Capacitated Cloudlet Placements in Wireless Metropolitan Area Networks

Zichuan Xu†, Weifa Liang†, Wenzheng Xu‡†, Mike Jia†, and Song Guo¶
†Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia
‡College of Computer Science, Sichuan University, Chengdu, 610065, P. R. China
¶School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan
Email: edward.xu@anu.edu.au, wliang@cs.anu.edu.au, wenzheng.xu3@gmail.com, u5515287@anu.edu.au, sguo@u-aizu.ac.jp

***Abstract*—In this paper we study the cloudlet placement problem in a large-scale Wireless Metropolitan Area Network (WMAN) that consists of many wireless Access Points (APs). Although most existing studies in mobile cloud computing mainly focus on energy savings of mobile devices by offloading computing-intensive jobs from them to remote clouds, the access delay between mobile users and the clouds usually is large and sometimes unbearable. Cloudlet as a new technology is capable to bridge this gap, and has been demonstrated to enhance the performance of mobile devices significantly while meeting the crisp response time requirements of mobile users. In this paper we consider placing multiple cloudlets with different computing capacities at some strategic local locations in a WMAN to reduce the average cloudlet access delay of mobile users at different APs. We first formulate this problem as a novel capacitated cloudlet placement problem that places K cloudlets to some locations in the WMAN with the objective to minimize the average cloudlet access delay between the mobile users and the cloudlets serving their requests. We then propose a fast yet efficient heuristic. For a special case of the problem where all cloudlets have the identical computing capacity, we devise a novel approximation algorithm with a guaranteed approximation ratio. In addition, We also consider allocating user requests to cloudlets by devising an efficient online algorithm for such an assignment. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are promising and scalable.**

I. Introduction

In recent years, mobile devices have undergone a transformation from bulky gadgets with limited functionalities to indispensable everyday accessories. Advances in mobile hardware technology have led to an explosive growth in mobile application markets. Although mobile applications are becoming increasingly computational-intensive, the computing capacity of mobile devices remains limited, due to the considerations of weight, size, battery life, ergonomics, and heat dissipation of portable mobile devices [17]. A powerful approach to enhancing the performance of mobile applications is enabling mobile devices to access rich resources in remote clouds, by mitigating the workload on the mobile devices [10], [22].

Although clouds have rich computing and storage resources, they typically are far away from mobile users. Communication delays between the clouds and their mobile users thus can be long and unpredictable. This is problematic for these mobile applications in which a crisp response time is critical to their users, such as augmented reality, speech recognition, navigation, language translation, etc [17], [19]. To reduce this long access delay, cloudlets were proposed as an alternative solution [17] to powerful clouds. Cloudlets are resource-rich server clusters co-located with wireless Access Points (APs) in a local network, and mobile users can offload their tasks to local cloudlets for processing [5], [17].As cloudlets are self-managing, with few requirements other than power and Internet connectivity, they can be deployed in existing networks, leading them to be viewed as 'data-centers in a box'. The physical proximity between mobile users and cloudlets means that the cloudlet access delay on task offloading is greatly reduced, compared to remote clouds, thereby significantly improving user experiences.

Most existing studies focused mainly on offloading tasks of mobile users to cloudlets for energy savings of mobile devices, assuming that the cloudlets have already been placed [4], [6], [10], [11], [22]. Little attention has ever been paid to cloudlet placements and the impact of different placements on mobile users. The cloudlet placement is imperative to fully utilize the capacity of cloudlets and thus reduce the average cloudlet access delay of mobile users. The locations of cloudlets are critical to the delay tolerance of mobile users, especially in a large-scale Wireless Metropolitan Area Network (WMAN) that consists of hundreds, even thousands of Access Points (APs), where mobile users access the cloudlets through their local APs. Due to the large size of the WMAN, poor cloudlet placements will result in a long access delays between mobile users and their cloudlets, and the load imbalance among cloudlets, i.e., some of the cloudlets are overloaded while others are underutilized and even idle. Therefore, strategic placement of cloudlets in a WMAN will significantly improve the performance of various mobile applications such as the average cloudlet access delay, etc.

In this paper, we focus on the cloudlet placement problem in a large-scale WMAN deployed in a metropolitan area, where a service provider (usually the local government) is planning to deploy K cloudlets with different computing capacities at some strategic AP locations for mobile user access, the objective is to minimize the average cloudlet access delay between mobile users and the cloudlets serving the user requests. To the best of our knowledge, we are the first to tackle this cloudlet placement problem. The challenge associated with such placements are that given $K \geq 1$ cloudlets to be placed in a large-scale WMAN, which cloudlets should be placed to which locations, and which user requests should be assigned to which cloudlets so that the average cloudlet access delay among the mobile users is minimized? As the problem considered is NP-hard, is there an approximation algorithm with a guaranteed approximation ratio for it? In this paper we will address the challenges and answer the question.

There are several placement problems including cache placements and server placements that have been extensively

978-1-4673-6770-7/15/$31.00 ©2015 IEEE

studied in the past [15], [21]. For example, the cache placement problem is to choose K replicas or hosting services among N potential sites with identical capacities, such that the latency experienced by users is minimized [21], which usually is reduced to the capacitated K-median problem. Due to the NP-hardness of the latter, there are approximation algorithms for unsplittable and splittable versions of the problem [3], [14], where 'unsplittable' refers to that a user request can be served by only one center [3], and 'splittable' indicates that the user request can be served by multiple centers [14].

Although there are some similarities between cache or sever placement problem [15], [21] and the cloudlet placement problem studied in the paper, there are two essential differences between them. First, existing studies assumed that either there is no capacity constraint on each cache server, or the capacity of each cache server is identical, while we consider cloudlets with computing capacity constraint. Second, the problem scale is different, and there may be a huge number of user requests in a large-scale WMAN, e.g., several millions in a metropolitan city, since there are increasingly a large number of mobile users. To simply apply an existing algorithm [3] to solve the cloudlet placement problem, we must divide the user requests at each AP into many virtual APs with each virtual AP corresponding to one user request and then apply the algorithm. Therefore, the existing algorithm in [3] is not applicable to the problem of concern in this paper due to its poor scalability. New algorithm must be devised to deal with a large number of user requests. We will address this scalability by applying a novel scaling technique, which can significantly reduce the time complexity of the proposed algorithm, while the quality of the solution is still guaranteed.

The main contributions of this paper are as follows. We first study multiple cloudlet placements with different computing capacities in a large-scale WMAN deployed in a metropolitan region, by formulating a novel capacitated cloudlet placement problem with the objective of minimizing the average cloudlet access delay between mobile users and the cloudlets serving their requests. We then show that the problem is NP-hard, and propose a fast, scalable heuristic solution. In addition, we also devise a novel approximation algorithm with a guaranteed approximation ratio for a special case of the problem where all cloudlets have identical computing capabilities. When all cloudlets have been placed, we propose an efficient online algorithm for user request assignment to cloudlets. We finally evaluate the performance of the proposed algorithms through experimental simulation. The simulation results demonstrate that the proposed algorithms are very promising.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces the system model and problem definitions. Section IV provides a fast yet scalable heuristic, and section V devises an approximation algorithm if all cloudlets have identical capacities. Section VI devises an online algorithm for user request assignment. Section VII evaluates the performance of the proposed algorithms by experimental simulation, and Section VIII concludes the paper.

II. Related work

Although most studies in literature mainly focused on offloading mobile user tasks to remote clouds [10], [22], cloudlets have been quickly gaining recognition as alternative offloading destinations due to their short response time [4], [6], [8], [9], [12], [11], [16], [18], [20]. For example, the system Odessa [16] was designed to enable interactive mobile applications while satisfying crisp response time requirements of the applications. Hoang *et al.* [8] proposed a linear programming solution for the task offloading problem by considering the QoS requirements of mobile users with an aim to maximize the revenue of a service provider. Xia *et al.* [20] devised a novel online algorithm for admitting user requests to a cloudlet dynamically.

Despite the increasing momentum of cloudlet research in Mobile Computing, the problem of cloudlet placement within a network has largely been overlooked. Previous works typically assumed that cloudlets are used in small private WLANs such as in campuses, buildings, or even at office floors. In such a setting, it can be argued that the placement of cloudlets is trivial. Wherever the cloudlet is placed, the small network size implies that the communication delay between the cloudlets and their users is negligible. However, the cloudlet placement in large-scale WMANs is non-trivial, as WMANs typically covers large cities, and can have millions of users with hundreds, even thousands of APs. In such a WMAN, a user could be a significant number of hops away from its nearest cloudlet. Thus, the average cloudlet access delay between a user and the cloudlet serving the user can adversely affect the experience of using the service. As a result, placing cloudlets with different processing capacities to some strategic locations in a WMAN is imperative to minimize the average cloudlet access delay and fully utilize the capability of the cloudlets.

III. Preliminaries

In this section we introduce the system model, followed by defining the problem precisely.

A. System model

We focus on the cloudlet placements in a WMAN that is a computer network providing wireless Internet coverage for mobile users in a large-scale metropolitan area. The network is often owned and operated by the local government as public infrastructures [13]. We believe cloudlets are particularly suitable for the WMAN. Firstly, metropolitan areas have a high population density, meaning that cloudlets will be accessible by a large number of mobile users. This improves the cost-effectiveness of cloudlet deployment as they are less likely to be idle. Secondly, due to the size of the network, the service provider can take advantage of economics of the scale when offering cloudlet services through the WMAN, making cloudlet services more affordable to the general public.

We consider a WMAN that consists of many wireless APs at various strategic locations, where mobile users can access public computing and storage resources in cloudlets located at a set S of locations through their nearby APs. The network thus can be represented by a connected, undirected graph $G = (V \cup S, E)$, where V is the set of APs and S is the set of potential locations of cloudlets, and E is the set of links between two APs in V or between an AP and a cloudlet located at a location in S. Let $n = |V|$ and $m = |E|$. Assume that $|S| \leq |V|$. In practice, some of the cloudlets usually are co-located with the APs, i.e., $S \subseteq V$. There is a high-speed link in E or an optical routing path between two APs or an AP and a cloudlet.

The weight of the link represents the data transmission latency between the its two endpoints (APs), and the latency between an AP and a cloudlet is the accumulative latency of all edges of the shortest path from the AP to the cloudlet. Associated with each node in V, there is an integer weight $w(v)$ (> 0), representing *the expected number of user requests* using the AP to access cloudlets in the network. As APs usually are deployed at strategic locations such as shopping malls, train stations, schools, and libraries, the expected number of user requests $w(v)$ per unit time at each AP v can be estimated by its population density in that area, or can be derived from historic AP access information.

Assume that there are K cloudlets with computing capacities $c_1, c_2, \ldots, c_K$ respectively to be placed to the K locations in S. For the sake of simplicity, we assume that the cloudlets are co-located with some of the APs, i.e., $S \subseteq V$. We further assume that $K \ll |S| \leq |V|$. As the processing ability of a cloudlet is typically proportional to its computing capacity, the number of user requests admitted by the cloudlet at any moment is proportional to its computing capacity. For the sake of convenience, we here assume that each user request on average takes one unit cloudlet capacity (e.g., virtual machine (VM)) to process, where a user may request multiple VMs for his/her tasks of an application [17]. Therefore, the assumption that each request takes one unit cloudlet capacity can be easily applied to various application scenarios.

Given the K cloudlets co-located with some locations in S, mobile users can offload their tasks to the cloudlets through their local APs. If a cloudlet is co-located with an AP, this will lead to the minimum cloudlet access delay of users located at that AP; otherwise, the mobile user requests will be relayed to nearby cloudlets for further processing, this will result in a cloudlet access delay due to the accumulative delay of multiple hop relays. Fig. 1 illustrates a WMAN network.

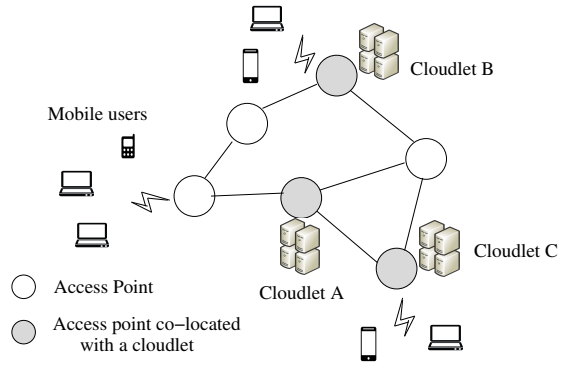


Fig. 1. A WMAN $G = (V, E)$ with $K = 3$ cloudlets.

B. Problem Definition

The capacitated cloudlet placement problem in a WMAN $G(V \cup S, E)$ is thus defined as follows. Given the network G, the expected number of user requests at each AP v_j which is a weight function $w : V \mapsto \mathbb{N}$, data transmission latencies on links $d : E \mapsto \mathbb{R}^{\geq 0}$, K (≥ 1) cloudlets with capacities $c_1, c_2, \ldots, c_K$ respectively, and a set S of potential locations for the K cloudlets with $S \subseteq V$, the problem is to place the K cloudlets to K locations in a configuration of S such that the average cloudlet access delay between mobile users and the cloudlets serving their requests is minimized, subject to that the number of user requests serviced by each cloudlet is no more than its processing capacity, where the cloudlet access delay of a user request d_{jl} is the length of the shortest path from its AP $v_j \in V$ to a cloudlet placed at location $v_l \in S$ serving the request. In other words, the problem is to identify K locations from $|S|$ ($\geq K$) potential locations and place the K cloudlets and to determine at which location the cloudlet C_i with capacity c_i should be placed such that the average cloudlet access delay of the user requests from their APs to their allocated cloudlets is minimized for all i with $1 \leq i \leq K$.

The capacitated cloudlet placement problem is NP-hard by a simple reduction from another NP-hard problem – *the metric K-median problem* [3].

IV. Algorithm for the Capacitated Cloudlet Placement Problem

Due to the NP-hardness of the capacitated cloudlet placement problem, we here devise an efficient heuristic for it.

We assume that the K cloudlets with different computing capabilities have been sorted in decreasing order of capacity, i.e., $c_1 \geq c_2 \geq \ldots \geq c_K$. The proposed algorithm proceeds iteratively. The K cloudlets are placed at K locations of the $|S|$ potential locations one by one, according to the decreasing order of their capacities. Specifically, within iteration i, cloudlets $C_1, C_2, \ldots, C_{i-1}$ have already been placed, and cloudlet C_i will be placed. Let $v_{l(i)}$ be the found location of cloudlet C_i. It should not have been occupied by any placed cloudlet, which implies that each potential placement location of cloudlet C_i should be chosen from $S \setminus \{v_{l(1)}, v_{l(2)}, \ldots, v_{l(i-1)}\}$. Also, this location $v_{l(i)}$ for C_i needs to 'cover' as many user requests from APs, while achieving the minimum average cloudlet access delay of all the user requests covered by the location. To find such a location for cloudlet C_i, determining which APs should send their user requests to C_i is crucial, as requests from different APs have different access delays. Our basic idea is to place cloudlet C_i to a location that has the minimum cloudlet access delay for all requests that will be routed at the location. Notice that the user requests $w(v_j)$ at each AP v_j may be sent to multiple cloudlets if there is no enough computational capacity left in C_i for processing all requests at AP v_j, which means that at each iteration of the algorithm there may exist not-fully-covered APs (not all user requests from an AP have been assigned to the first $(i-1)$ placed cloudlets). We sort all not-fully-covered APs in increasing order of the delays between these APs and the potential placement location v_l of cloudlet C_i, i.e., d_{lj} for each such an AP v_j. The first least r APs with the total number of remaining user requests exceeding the capacity c_i of C_i is identified. All the requests from these first $r-1$ APs and some of the requests from the rth AP will be sent to cloudlet C_i. That is, the number of user requests packed by C_i is equal to its capacity c_i. The detailed algorithm is described in `Algorithm` 1.

Theorem 1: Given a WMAN $G = (V \cup S, E)$, the expected number of user requests $w(v)$ at each AP $v \in V$, and K cloudlets $C_1, C_2, \ldots, C_K$ with processing capacities $c_1, c_2, \ldots, c_K$, respectively, there is a fast, scalable algorithm for the capacitated cloudlet placement problem, which takes $O(Kn^2 \log n + nm)$ time, assuming that $w(v) \leq$

Algorithm 1 `Greedy_Heuristic`

Input: WMAN network $G = (V \cup S, E)$, the number $w(v_j)$ of user requests at each AP $v_j \in V$, the communication delay of each edge in E, K cloudlets with given capacities $c_1, c_2, \ldots$, and c_K respectively, and the set S ($\subseteq V$) of potential locations for the K cloudlets.
Output: the placement locations of the K cloudlets.

$U \leftarrow S$; /* all potential locations of cloudlet placements*/
$L \leftarrow \emptyset$; /* the set of cloudlets that have been placed*/
Compute all pairs of shortest paths for each pair of APs in the network;
Sort the cloudlet capacity in decreasing order, i.e., $c_1 \geq c_2 \geq \ldots \geq c_K$;
for $i \leftarrow 1$ to K **do**
  /* place the cloudlet with capacity c_i to one location in U */
  $delay_{min} \leftarrow \infty$;
  $l(i) \leftarrow 0$; /* the location for cloudlet C_i */
  $AP\_cloudlet_i \leftarrow \emptyset$; /* the set of APs whose user requests are already been allocated to C_i */
  for each location $v_l \in U$ **do**
    /* if cloudlet C_i with capacity c_i is placed in location v_l, identify a subset of APs, $AP\_cloudlet_{il}$ from $V \setminus AP\_cloudlet$ that have not been assigned to any cloudlet yet */
    $AP\_cloudlet_{il} \leftarrow \emptyset$;
    Sort all not-fully-covered APs in increasing order of the access delay between the APs and the potential location v_l of cloudlet C_i, i.e., d_{lj} for each AP $v_j \in V \setminus AP\_cloudlet$;
    Find the least r APs in the sorted AP sequence such that the total number of user requests of these r APs exceeds the capacity of C_i;
    Add all the first $r-1$ APs to $AP\_cloudlet_{il}$, meaning that the user requests from these APs are routed to C_i;
    Route a portion z_{ir} of the user requests $w(v_r)$ of the rth AP to C_i, where $z_{ir} = c_i - \sum_{v_j \in AP\_cloudlet_{il}} w(v_j)$;
    $w(v_r) \leftarrow w(v_r) - z_{ir}$ /* remaining user requests at the rth AP*/;
    $Delay_{il} \leftarrow \sum_{v_j \in AP\_cloudlet_{il}} d_{lj} w(v_j) + d_{lr} z_{ir}$;
    if $Delay_{il} < delay_{min}$ **then**
      $l(i) \leftarrow l$, and $delay_{min} \leftarrow Delay_{il}$;
      $AP\_cloudlet_i \leftarrow AP\_cloudlet_{il}$;
  C_i is placed at location $v_{l(i)}$ and all user requests from each $AP \in AP\_cloudlet_i$ will be covered;
  $L \leftarrow L \cup \{< v_{l(i)}, C_i >\}$, $U \leftarrow U \setminus \{v_{l(i)}\}$;

$\min_{1 \leq i \leq K}\{c_i\}$ for any $v \in V$ and $K \leq |S| \leq |V|$, where $n = |V|$ and $m = |E|$.

Proof: The analysis of the time complexity is straightforward, omitted. ■

V. Approximation Algorithm for a Special Capacitated Cloudlet Placement Problem

In this section, we deal with a special case of the problem where all the K cloudlets have identical capacities c (i.e., $c_1 = c_2 = \cdots = c_K = c$) and $S = V$. This special case is still NP-hard, which can be reduced from *the capacitated K-median problem*, in which each AP has only one user request [3], through dividing the user requests at each AP into many virtual APs with each virtual AP having one user request. This however suffers from poor scalability, due to the large numer of user requests at each AP. To address this issue, we first introduce the capacitated K-median problem, and then devise an approximation algorithm by applying a non-trivial scaling technique.

A. The capacitated K-median problem

Given a set of locations V_U with each location $j \in V_U$ having a demand $w_j \geq 0$ to be served, K centers with each having a service capacity M, and the service cost d_{ij} for a center at location i serving one unit of demand from location j, the capacitated K-median problem is to find K different locations in V_U to place the K centers and allocate the demand w_j of each location $j \in V_U$ to a placed center such that the total service cost is minimized, subject to that the total demand served by each center is no more than its capacity M and the demand of each location $j \in V_U$ must be served by one center only, where the service costs are non-negative, symmetric, and satisfy the triangle inequality.

Theorem 2: [3] There is an approximation algorithm for the capacitated K-median problem in a metric space, which delivers an approximate solution with an approximation ratio 16 in the service cost, while the total demand served by each center is no more than 4 times of the capacity M of the center.

B. Approximation algorithm

We now devise an approximation algorithm for this special capacitated cloudlet placement problem. The strategy we adopt is to reduce the problem to the capacitated K-median problem, an approximate solution to the latter will form a base for the approximate solution to the problem through a proper transformation. It can be seen from Theorem 2 that if the number of user requests from all APs is polynomial of the number of APs n in the WMAN, the special capacitated cloudlet placement problem can be reduced to the capacitated K-median problem, by replicating the $w(v_j)$ user requests at each AP v_j into $w(v_j)$ virtual nodes with each virtual node corresponding to a user request. An approximate solution to the latter in turn returns an approximate solution to the former. Otherwise (e.g., the number of user requests is exponential of the number of APs n), the running time of this method may not be polynomial. Let N_0 be a basic unit of the number of user requests, where N_0 is an integer with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$ and $N_0 \leq c$. Thus, each AP $v_j \in V$ with $w(v_j)$ expected user requests will have $n_j = \lfloor w(v_j)/N_0 \rfloor$ units. Similarly, each cloudlet C_i will have a capacity of $c' = \lceil c/N_0 \rceil$ units. By adopting an appropriate value N_0, the total number of units $\sum_{v_j \in V} n_j$ for all user requests in the system becomes polynomial in terms of the number of APs n.

Given the original graph $G = (V, E)$, the expected number of user requests at each AP $w : V \mapsto \mathbb{N}$, the data transmission delay between every two APs $d : E \mapsto \mathbb{R}^{\geq 0}$, and the K cloudlets with each having a capacity c, we construct an auxiliary complete graph $G_U = (V_U, E_U)$ as follows. For each AP v_j with $w(v_j)$ expected user requests, we add $n_j = \lfloor w(v_j)/N_0 \rfloor$ virtual nodes $v_j^1, v_j^2, \ldots, v_j^{n_j}$ into V_U with each serving as a source node and having the demand of 1. There is an edge e in E_U for each pair of virtual nodes v_i^x and v_j^y in V_U. The weight of edge $e = (v_i^x, v_j^y) \in E_U$ is the delay between APs v_i and v_j in G for transmitting N_0 user requests (i.e., $N_0 d_{ij}$) between them if $v_i \neq v_j$, and 0 otherwise.

The problem now is to place the K cloudlets with identical capacities c_i' in V_U to cover all source nodes such that the weighted sum of the edges between placed cloudlets and their covered source nodes is minimized. This is *the capacitated K-median problem*, which can be solved by an approximation algorithm due to Charikar *et al.* [3]. Having the approximate solution, a feasible solution to the special capacitated cloudlet placement problem then can be obtained. That is, the K cloudlets will be placed at the K locations in the approximate solution. For each AP v_j with $n_j = \lfloor w(v_j)/N_0 \rfloor$ virtual nodes $v_j^1, v_j^2, \ldots, v_j^{n_j}$ in G_U, assume that v_j^l is allocated to a cloudlet deployed at location i in the approximate solution, then N_0

user requests at AP v_j will be assigned to the cloudlet at location i, where $1 \leq l \leq n_j$. The remaining $w(v_j) - N_0 \cdot n_j$ user requests at v_j will be assigned to a nearest cloudlet with the minimum accumulative delay among the cloudlets to which the virtual nodes $v_j^1, v_j^2, \ldots, v_j^{n_j}$ are allocated. The detailed algorithm description is given in `Algorithm` 2.

Algorithm 2 `Appro`

Input: a WMAN $G = (V, E)$, the number $w(v_j)$ of user requests at each AP $v_j \in V$, the delay $d(e)$ of each edge $e \in E$, K cloudlets with a uniform capacity c, and a positive integer N_0 with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$ and $N_0 \leq c$.

Output: The placement locations of the K cloudlets.

1: Construct an auxiliary graph $G_U = (V_U, E_U)$ from G, where $n_j = \lfloor w(v_j)/N_0 \rfloor$ virtual nodes are added to V_U and each of them is treated as a 'source node' with demand of 1 for each AP $v_j \in V$ with $w(v_j)$ user requests, and an edge e between any two virtual nodes v_i^x and v_j^y in V_U is added to E_U and its weight is $N_0 d_{ij}$ if $v_i \neq v_j$ and 0 otherwise;
2: Find an approximate solution for the capacitated K-median problem in $G_U(V_u, E_U)$, by applying the algorithm due to Charikar *et al.* [3];
3: Place the K cloudlets to the locations in the approximate solution;
4: For each virtual node v_j^l of AP v_j, assign N_0 user requests at AP v_j to the cloudlet to which the node v_j^l is allocated in the approximate solution;
5: Assign the rest $w(v_j) - N_0 n_j$ user requests at AP v_j to a nearest cloudlet among the cloudlets to which the n_j virtual nodes of AP v_j are allocated.

C. Algorithm Analysis

We analyze the approximation ratio of the proposed approximation algorithm `Appro` through Theorems 3 and 4.

Let $G' = (V', E'; d')$ be another graph that is identical to the original graph $G = (V, E; d)$, i.e., $V' = V$, $E' = E$, and $d'_{ij} = d_{ij}$ for any two APs v_i and v_j in V. However, assume that the number of user requests at each AP $v_{j'}$ in G' is $\lfloor w(v_j)/N_0 \rfloor N_0$, which is no more than the number of user requests $w(v_j)$ at AP v_j in G, i.e., $\lfloor w(v_j)/N_0 \rfloor N_0 \leq w(v_j)$. We then deploy the K cloudlets in network G' while the capacity of each cloudlet C_i now is $\lceil \frac{c}{N_0} \rceil N_0$, not c. It is obvious that $\lceil \frac{c}{N_0} \rceil N_0 \geq c$. For each AP $v_{j'}$ in G', we can split its $\lfloor w(v_j)/N_0 \rfloor N_0$ user requests into $n_j = \lfloor w(v_j)/N_0 \rfloor$ blocks with each block containing N_0 requests exactly.

We consider two types of the special cloudlet placement problem in graph $G'(V', E')$: the first one is that each user request can be allocated to any cloudlet; and the second one is that the whole N_0 user requests of each block must be allocated to a single cloudlet but the user requests in different blocks can be allocated to different cloudlets. In the following we show that the costs of the optimal solutions to these two different types of cloudlet placement problems are equal.

Theorem 3: Denote by OPT_1 and OPT_2 the costs of the optimal solutions to these two types of the special cloudlet placement problem, respectively, we have $OPT_1 = OPT_2$.

Proof: It can be seen that $OPT_1 \leq OPT_2$, since the optimal solution to the second type of the special cloudlet placement problem is a feasible solution to the first type one. The rest is to show that $OPT_2 \leq OPT_1$.

Denote by $\mathbb{X}_1$ and $\mathbb{X}_2$ the optimal solutions to the first and second types of the special cloudlet placement problem, respectively. Assume that the K cloudlets are placed at locations $i_1, i_2, \cdots, i_K$ in solution $\mathbb{X}_1$. Recall that the $\lfloor w(v_j)/N_0 \rfloor N_0$ user requests at AP $v_{j'}$ have been split into $n_j = \lfloor w(v_j)/N_0 \rfloor$ blocks with each block containing N_0 requests exactly. Let $n_r = \sum_{v_j \in V} n_j$ be the total number of blocks in G' that can be represented by $b_1, b_2, \cdots, b_{n_r}$, respectively. Let $B = \{b_1, b_2, \cdots, b_{n_r}\}$ be the set of the n_r blocks and $I = \{i_1, i_2, \cdots, i_K\}$ the set of the K cloudlet locations in solution $\mathbb{X}_1$.

We construct an auxiliary flow graph $G_f = (\{s\} \cup B \cup I \cup \{t\}, E_f)$ from the n_r blocks in B and the K locations in I as follows. There is a directed edge in E_f from source s to each block $b_j \in B$ with a capacity of 1 and a cost $d''_{sj} = 0$. For each block $b_j \in B$ and each location $i_l \in I$, there is a directed edge in E_f from b_j to i_l with a capacity of 1 and a cost $d''_{jl} = d_{jl} N_0$ (i.e., the total delay of transmitting the N_0 user requests from block b_j to the cloudlet located at i_l). Furthermore, there is an edge in E_f from each location $i_l \in I$ to sink t with a capacity of $\lceil \frac{c}{N_0} \rceil$ units and a cost $d''_{lt} = 0$.

Given a flow f from s to t in graph $G_f(\{s\} \cup B \cup I \cup \{t\}, E_f)$, the cost of the flow is defined as $\sum_{e \in E_f} f_e \cdot d''_e$. Following the assumption that the number of user requests in G is no more than the total capacity of the K cloudlets, i.e., $\sum_{v_j \in V} w(v_j) \leq K \cdot c$, we then have $n_r = \sum_{v_j \in V} \lfloor w(v_j)/N_0 \rfloor \leq K \lceil \frac{c}{N_0} \rceil$, and the value of a maximum flow in G_f from s to t is n_r.

Consider *the minimum cost maximum flow problem* in G_f that is to find a maximum flow from s to t in G_f with the minimum cost [1]. From the optimal solution $\mathbb{X}_1$ to the first type of the special cloudlet placement problem, we can construct a fractional maximum flow f (not necessarily having the minimum cost of the flow) to the minimum cost maximum flow problem in G_f, i.e., for the N_0 user requests in each block b_j, assume that there are x_{ji} user requests of the N_0 requests allocated to the cloudlet located at $i \in I$, the fractional flow f_{ji} from block b_j to location i is $\frac{x_{ji}}{N_0}$. It can easily be verified that the cost of this fractional flow is OPT_1. On the other hand, notice that the capacity of each edge in graph G_f is integral. Following the well-known integrality property for the minimum cost maximum flow problem [1], there is an integral minimum cost maximum flow f^* for the problem. That is, for each block $b_j \in B$ and each location $i_l \in I$, the flow f^*_{jl} from b_j to i_l is either 0 or 1 as the capacity of edge (b_j, i_l) is 1. Denote by $D(f^*)$ the cost of flow f^*, i.e., $D(f^*) = \sum_{e \in E_f} f^*_e \cdot d''_e$. Then, $D(f^*) \leq OPT_1$, since the solution $\mathbb{X}_1$ is a feasible solution to the minimum cost maximum flow problem in G_f. Also, we know that this integral maximum flow f^* corresponds to a feasible solution with cost $D(f^*)$ to the second type of the special cloudlet placement problem. As $\mathbb{X}_2$ with cost OPT_2 is an optimal solution to the problem, then $OPT_2 \leq D(f^*)$. Therefore, we have $OPT_2 \leq D(f^*) \leq OPT_1$. By combining the above discussions, we have $OPT_2 = OPT_1$. ■

Having Theorem 3, we are ready to analyze the approximation ratio of `Algorithm` 2

Theorem 4: Given a WMAN $G = (V, E)$ with each AP $v_j \in V$ having $w(v_j)$ user requests and K cloudlets with identical capacity of c, there is an approximation algorithm for this special cloudlet placement problem, which delivers an approximate solution with an approximation ratio of $16(1+\epsilon)$, while the number of user requests served by each cloudlet is no more than $8(1+\delta)c$, where $\epsilon = \max_{v_j \in V}\{\frac{1}{\lfloor w(v_j)/N_0 \rfloor}\} \leq 1$, $\delta = \frac{N_0}{c} \leq 1$, N_0 is a positive integer with $1 \leq N_0 \leq \min_{v_j \in V}\{w(v_j)\}$, and $N_0 \leq c$.

Proof: We first analyze the approximation ratio of `Algorithm 2` as follows. Denote by OPT the optimal (or minimum) total delay for the special capacitated cloudlet placement problem with each cloudlet having identical capacity in the original graph $G(V, E)$. Denote by OPT_U the optimal total service cost for the capacitated K-median problem in the auxiliary graph $G_U(V_U, E_U)$. Following the construction of G_U and Theorem 3, OPT_U is no more than OPT as the number of user requests n_jN_0 at each AP v_j in G_U is no more than the number of user requests $w(v_j)$ at AP v_j in G, i.e., $\lfloor w(v_j)/N_0 \rfloor N_0 \leq w(v_j)$, and the capacity $\lceil \frac{c}{N_0} \rceil N_0$ of each cloudlet in G_U is no less than the capacity c of the cloudlet in G, i.e., $\lceil \frac{c}{N_0} \rceil N_0 \geq c$. Following Theorem 2, the cost (or the total delay) of the approximate solution delivered by the approximation algorithm for the capacitated K-median problem in graph G_U is no more than $16 \cdot OPT_U$. Denote by D_j the total delay incurred by the assigning the n_jN_0 user requests at AP v_j to their allocated cloudlets in the approximate solution. Then, $\sum_{v_j \in V} D_j \leq 16 \cdot OPT_U$. Since the rest $w(v_j) - n_jN_0$ ($\leq N_0$) user requests at AP v_j are assigned to the nearest cloudlet among the cloudlets to which the n_jN_0 user requests are allocated, the total delay incurred by these $w(v_j) - n_jN_0$ user requests is no more than $\frac{D_j}{n_jN_0}N_0 = \frac{D_j}{n_j}$. Therefore, the total delay of assigning all user requests to the K cloudlets is no more than $\sum_{v_j \in V}(D_j + \frac{D_j}{n_j}) = \sum_{v_j \in V} D_j(1 + \frac{1}{n_j}) \leq (1+\epsilon)\sum_{v_j \in V} D_j \leq (1+\epsilon)16 \cdot OPT_U \leq (1+\epsilon)16 \cdot OPT$, where $\epsilon = \max_{v_j \in V}\{\frac{1}{n_j}\} = \max_{v_j \in V}\{\frac{1}{\lfloor w(v_j)/N_0 \rfloor}\} \leq 1$.

We then show that the number of user requests served by each cloudlet in the solution delivered by `Algorithm 2` is no more than $8(1+\delta)c$, where $\delta = \frac{N_0}{c} \leq 1$. Following Theorem 2, the number of user requests allocated to each cloudlet is no more than $4\lceil \frac{c}{N_0} \rceil N_0 \leq 4(\frac{c}{N_0}+1)N_0 = 4(1+\frac{N_0}{c})c = 4(1+\delta)c$ in the approximate solution delivered by the approximation algorithm in [3]. We show that after having assigned the rest $w(v_j) - n_jN_0$ ($\leq N_0$) user requests from each AP v_j to the cloudlets, the number of user requests served by each cloudlet is no more than twice the number prior to this assignment. Assume that for a deployed cloudlet C_i, it processes user requests from APs $v_1, v_2, \cdots, v_p$ before assigning the remaining user requests. Following `Algorithm 2`, cloudlet C_i will process no less than N_0 user requests from each of these p APs. Since there are no more than N_0 remaining user requests at each of the p APs, the number of user requests assigned to cloudlet C_i thus is no more than $8(1+\delta)c$. ■

Notice that if the capacity of each cloudlet is not allowed to be overloaded, we may set the capacity of each cloudlet to a value of $\frac{c}{8(1+\delta)}$ instead of c, then apply the approximation algorithm for this new capacity. Following Theorem 4, none of the cloudlet will violate its original capacity c.

VI. Online User Request Assignment

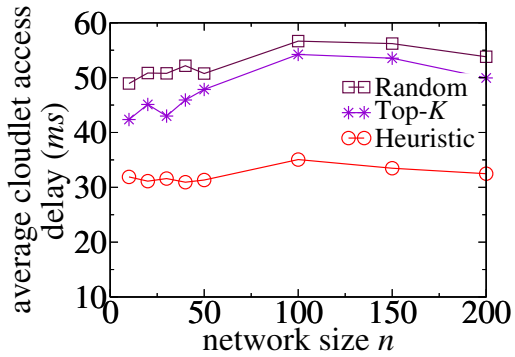
The proposed algorithm for the capacitated cloudlet placement problem so far assumed that the expected number of user requests of each AP is given, based on historic information. However, in reality, once the K cloudlets have been placed, the workloads of different cloudlets may be different over time, some of them may be overloaded while others are light-loaded or idle. On the other hand, the actual number of user requests from each AP may not be necessarily consistent with its expected number and is very likely to vary over time. Thus, there may have such situations where the actual number of user requests from some APs is much below their expected figures, while the actual number of user requests from the other APs may be above their original figures. It is obvious that dispatching much over than the expected number of user requests from an AP to its designated overloading cloudlet is not a smart choice. Instead, some of these requests can be assigned to other light-loaded cloudlets for processing. Consider such dynamic behaviors of both user requests and the workloads of the K cloudlets, in this section we deal with the online assignment problem of user requests to minimize the average cloudlet access delay of mobile users, assuming that the K cloudlets have already been placed. We assume that the time is slotted into equal time slots, and the admission of new user requests is taken in the beginning of each time slot.

The idea behind the proposed algorithm for the online assignment problem of user requests in network G is to reduce the problem to *the minimum-cost maximum flow problem* in another auxiliary flow network G' and the solution to the latter in turn will return a solution to the former. Specifically, the flow network G' is derived from the WMAN $G = (V, E)$ and the K cloudlet placement sites. For the sake of convenience, assume that the K cloudlets have been placed at AP locations $v_1, v_2, \ldots, v_K$, respectively, and the cloudlet C_i with capacity c_i is placed at location v_i, $1 \leq i \leq K$. Let $S = \{v_1, v_2, \ldots, v_K\}$ be the set of the locations of the placed K cloudlets. Assuming time is divided into equal time slots, there are $w(v_j, t)$ user requests that need to be assigned from AP v_j to the system in the beginning of time slot t, where $v_j \in V$. The network flow graph $G' = (\{a\} \cup V \cup S \cup \{b\}, E')$ is constructed from $G = (V, E)$ as follows. There is a 'virtual source' a and a 'virtual sink' b in G'. For each AP $v_j \in V$, there is a directed edge in E' from source a to v_j with a capacity of $w(v_j, t)$ (i.e., the number of user requests that needs to be scheduled at AP v_j at time slot t) and a cost of 0. For each AP $v_j \in V$ and each cloudlet $v_i \in S$, there is a directed edge in E' from v_j to v_i with a sufficiently large capacity and a cost of d_{ji} (i.e., the shortest delay of transmitting a user request from AP v_j to cloudlet v_i). Furthermore, for each cloudlet $v_i \in S$, there is a directed edge in E' from v_i to sink b with a capacity of $c'_i(t)$ (i.e., the residual processing capacity of cloudlet v_i at time slot t) and a cost of 0. Note that the capacity of each edge in network G' is integral. Following the well-known integrality property for the minimum-cost maximum flow problem [1], there is an integral minimum-cost maximum flow for the problem in G'. That is, the number of assigned user requests in the flow from each AP v_j to each cloudlet v_i is an integer. Such a user request assignment thus not only maintains load-balancing among the K cloudlets but also keeps the average cloudlet access delay of the user requests minimized. Also, there is a polynomial algorithm for the minimum-cost maximum flow problem [1]. As a result, we obtain an online algorithm for the request assignment problem, which is referred to as Algorithm `Online_Assignment`.

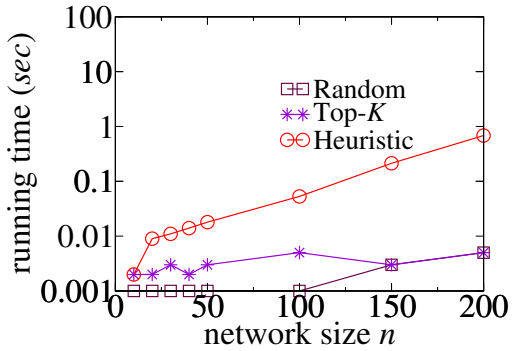
VII. Performance Evaluation

A. Experiment Settings

We assume that the WMAN $G(V, E)$ consists of 200 APs and there is an edge between every pair of nodes with a

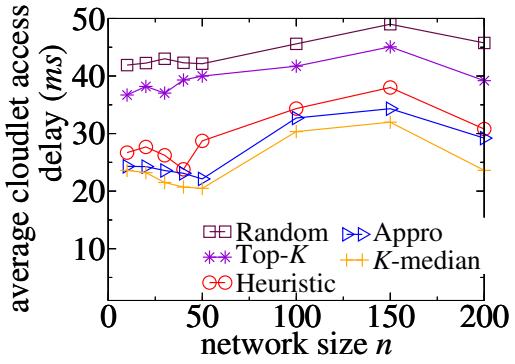


(a) Average cloudlet access delay

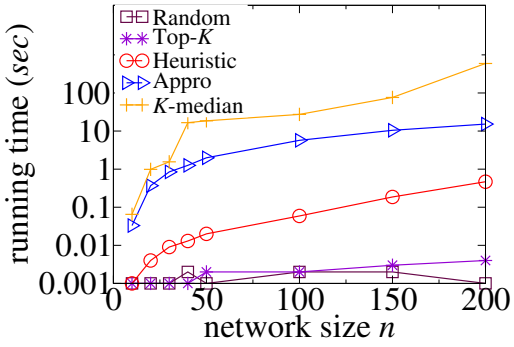


(b) Running time

Fig. 2. The performance of algorithms Heuristic, Random, and Top-K



(a) Average cloudlet access delay



(b) Running time

Fig. 3. The performance of algorithms Heuristic, Appro, Random, and Top-K with identical processing capacity

probability of 0.02. The network is generated by a popular tool GT-ITM [7]. We assume that the number of cloudlets is 10% of the number of APs in G, i.e., $K = 10\% \cdot 200 = 20$. Assume that S is the placement space of the K cloudlets and $S = V$. The expected number of user requests $w(v)$ at each AP $v \in V$ is randomly drawn from a value interval of [100, 1,000] [13]. Let R_{sum} be the total expected number of user requests, then $R_{sum} = \sum_{v \in V} w(v)$. The capacity of each cloudlet is randomly drawn in a value interval of [1,000, R_{sum}], and assume that the sum of capacities of the K cloudlets is no less than the total expected number of user requests R_{sum}. The delay of each link in G is a value randomly generated between 5*ms* and 50*ms* [17]. We also investigate the running time of the proposed algorithms, where the actual running time of each algorithm is the average of the running times of the algorithm for 15 times on a desktop with 2.66 GHz Intel Core 2 Quad CPU and 8GB RAM. Each value in figures is the mean of the results by applying each algorithm to 15 different combination of network topologies, edge weights, and the number of user requests at each AP. Unless otherwise specified, these default parameters will be adopted in our simulation.

We compare the performance of algorithms Heuristic and Appro against two heuristic algorithms: one is to place the K cloudlets to APs randomly; the other is to place the K cloudlets to the top-K APs, where an AP is a top-K AP if its number of user requests is one of the top-K values, which are referred to as algorithms Random and Top-K, respectively.

B. Performance Evaluation of Different Algorithms

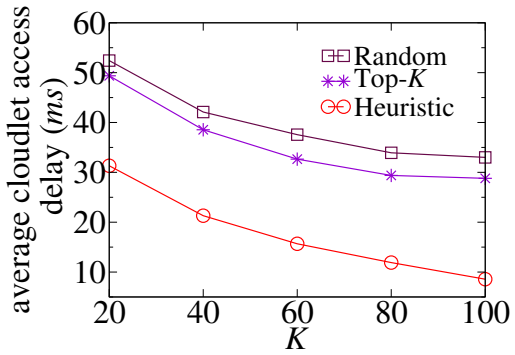
We first evaluate the performance of different algorithms. We investigate the performance of algorithm Heuristic against that of other algorithms Random, and Top-K by varying the number of APs n from 10 to 200 while fixing the ratio of the number of cloudlets to the number of APs at 0.1, i.e., $K/n = 0.1$. Fig. 2 (a) plots the curves of the average cloudlet access delays delivered by the algorithms, from which it can be seen that algorithm Heuristic significantly outperforms the other two algorithms Random and Top-K, and algorithm Top-K is only slightly better than algorithm Random. Specifically, the average cloudlet access delay delivered by algorithm Heuristic is less than those delivered by algorithms Random and Top-K by 25% and 30%, respectively.

We then study the scalability of different algorithms, by varying the number of APs n from 10 to 200, while keeping $\frac{K}{n} = 0.1$. Fig. 2 (b) illustrates the running time curves of algorithms Heuristic, Random, and Top-K, respectively, from which it can be seen that algorithm Heuristic exhibits excellent efficiency and scalability, which takes less than one second to find a feasible solution in a WMAN of 200 APs.

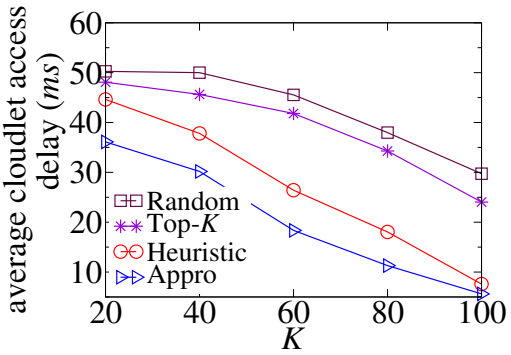
C. Performance Evaluation of the Approximation Algorithm

We also study the performance of algorithms Heuristic, Appro, Random, Top-K, and K-median when each cloudlet has an identical capacity, by varying the number of APs n from 10 to 200 while fixing $K/n = 0.1$.

Fig. 3 (a) depicts the average cloudlet access delay by algorithms Appro, Heuristic, Top-K, and Random. It can be seen from the figure that the average cloudlet access delay by algorithm Appro nearly approaches the optimal one, and the gap between them is only from 5% to 10%. Fig. 3 (a)



(a) Different cloudlet capacities



(b) Identical cloudlet capacities

Fig. 4. The impact of number of cloudlets K on the performance of algorithms Heuristic, Appro, Top-K, and Random.

clearly indicates that the performance of algorithm Appro is significantly better than those of algorithms Heuristic, Top-K, and Random. Specifically, the average cloudlet access delay by algorithm Appro is 10% to 20% less than that by algorithm Heuristic, and nearly 50% less than those by algorithms Top-K and Random. In terms of the running time, it can be seen from Fig. 3 (b) that algorithm Appro takes a few more minutes than that of the other algorithms to find a feasible solution. Although algorithm Random takes the least time to deliver a feasible solution, its solution is worse than other algorithms. In addition, algorithm Appro performs marginally worse than that of algorithm K-median in terms of the average cloudlet access delay (Fig. 3(a)), while it runs much faster than algorithm K-median. For example, when the network size $n = 200$, the running time of algorithm Appro is only about 15 seconds while the running time of K-median is as high as 600 seconds.

We then study the impact of the number of cloudlets K on the performance of mentioned algorithms Heuristic, Appro, Top-K, and Random, by varying K from 20 to 100 when fixing n at 200. Figures 4 (a) and 4 (b) illustrate the curves of the average cloudlet access delay of different algorithms with and without uniform capacity, from which it can be seen that the average cloudlet access delays in the solutions by algorithms Heuristic and Appro are much lower than those by algorithms Top-K and Random. Furthermore, it can be seen from Fig. 4 that the average cloudlet access delay by each mentioned algorithm decreases with the growth of K, since each request has more chances to be sent to its nearest cloudlet for processing, with the growth of the number of cloudlets.

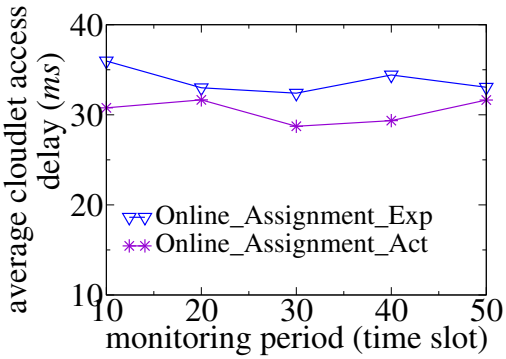
D. Performance Evaluation of the Online Assignment Algorithm

We finally investigate the performance of algorithm Online_Assignment by varying the monitoring periods. The actual number of requests at each AP within each time slot follows the following two distributions: (1) uniform distribution within an interval $[(1-\rho)w(v),\ (1+\rho)w(v)]$, where ρ is a constant with $0 < \rho < 1$ and its default value is set to 0.4 [2]; and (2) Zipf's distribution. Let $w_{sum}(t)$ be the total actual number of requests by all APs at time slot t, and assume that all APs are ranked in decreasing order of their expected numbers of requests. The actual number of requests of the AP i thus is $w_{sum}(t)\frac{1/i}{\sum_{j=1}^{|V|}1/j}$. In the following, we evaluate the performance of algorithm Online_Assignment of different monitoring periods, assuming that cloudlet placements are obtained with the knowledge of the expected and actual numbers of user requests, which are named by Online_Assignment_Exp and Online_Assignment_Act, respectively.

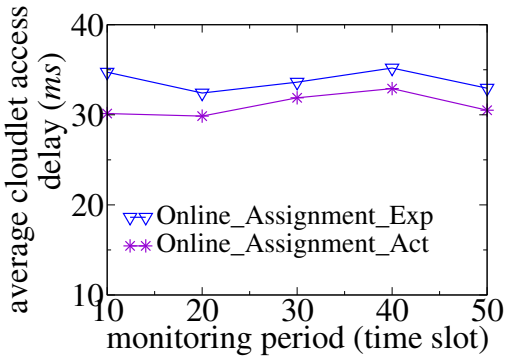
Figures 5(a) and 5(b) plot the curves of average cloudlet access delay by algorithms Online_Assignment_Exp, Online_Assignment_Act with actual number of user requests generated from the uniform distribution. We can see that the performance of algorithm Online_Assignment_Exp is only marginally worse than algorithm Online_Assignment_Act, and their performance gap is insignificant, with the increasing of the length of the monitoring period. For example, Fig. 5(a) shows that the average cloudlet access delay by algorithm Online_Assignment_Exp is only from 5% to 10% higher than that by algorithm Online_Assignment_Act. Although such gaps in Figures 5(c) and 5(d) are higher, the average cloudlet access delays are still around 10% better than the average performance of algorithm Top-K (Figures 2 and 3). Thus, Fig. 5 indicates that the cloudlet placements found by algorithms Heuristic and Appro based on the expected numbers of user requests are good enough to deal with the scenarios where the actual number of user requests are not consistent with the expected ones. Notice that the running time of the online request assignment is much smaller than that of K cloudlet placement the former is a subproblem of the latter, which takes around 110 milliseconds for a network size 200, compared with around 10 seconds for placing K cloudlets into the network by algorithm Appro. Due to the space limit, the running time of the online assignment algorithm is omitted.

VIII. Conclusion

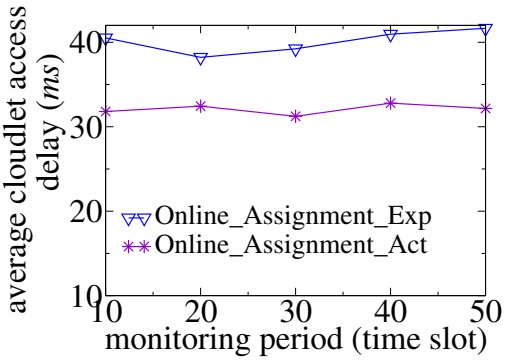
Cloudlets have been emerged as an important technology that can extend the computing capabilities significantly of resource-constrained mobile devices. In this paper we first studied the capacitated cloudlet placement problem in a large-scale Wireless Metropolitan Area Network with the objective to minimize the average cloudlet access delay between mobile users and the cloudlets serving their requests. We then proposed a fast, scalable heuristic solution. We also devised a novel approximation algorithm with a guaranteed approximation ratio for a special case of the problem where all cloudlets have identical computing capabilities. Furthermore, we proposed an efficient algorithm for user request assignment to cloudlets



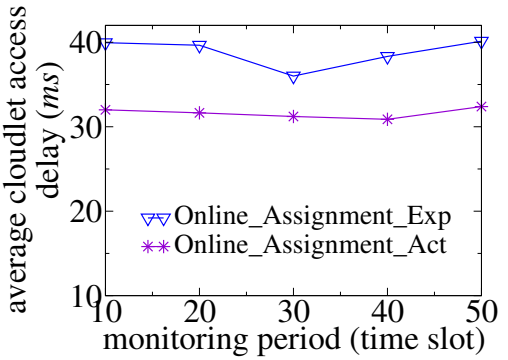
(a) Different cloudlet capacities and user request numbers following uniform distribution



(b) Identical cloudlet capacities and user request numbers following uniform distribution



(c) Different cloudlet capacities and user request numbers following Zipf distribution



(d) Identical cloudlet capacities and user request numbers following Zipf distribution

Fig. 5. Performance of algorithms `Online_Assignment_Exp` and `Online_Assignment_Act`

when all cloudlets have been placed. We finally evaluated the performance of the proposed algorithms by experimental simulations. The simulation results showed that the proposed algorithms are very promising.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.

[2] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan. Characterizing user behavior and network performance in a public wireless LAN. *Proc. of SIGMETRICS*, ACM, 2002.

[3] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Proc. of STOC*, ACM, 1999.

[4] B.G Chun *et. al.* Clonecloud: elastic execution between mobile device and cloud. *Proc. of EuroSys*, ACM, 2011.

[5] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan. How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users. *Proc. of PerCom*, IEEE, 2012.

[6] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: making smartphones last longer with code offload. *Proc. of MobiSys*, ACM, 2010.

[7] GT-ITM. *http://www.cc.gatech.edu/projects/gtitm/*.

[8] D. T. Hoang *et. al.* Optimal admission control policy for mobile cloud computing hotspot with cloudlet. *Proc. of WCNC*, IEEE, 2012.

[9] M. Jia, J. Cao, and W. Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. To appear in *IEEE Trans. on Cloud Computing*, 2015.

[10] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Cuckoo: a computation offloading framework for smartphones. *Mobile Computing, Applications, and Services*, Springer, pp.59–79, 2012.

[11] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. *Proc. of INFOCOM*, IEEE, 2012.

[12] K. Kumar and Y. Lu. Cloud computing for mobile users: can offloading computation save energy. *Journal of Computer*, Vol. 43, pp.51–56, 2010.

[13] LAN/MAN Standards Committee of the IEEE Computer Society. *IEEE standard for local and metropolitan area networks: overview and architecture*, IEEE Std 802-2001, Feb., 2002.

[14] S. Li. An improved approximation algorithm for the hard uniform capacitated k-median problem. *http://arxiv.org/abs/1406.4454*, 2014.

[15] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. *Proc. of INFOCOM*, IEEE, 2001.

[16] M.R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan. Odessa: enabling interactive perception applications on mobile devices. *Proc. of MobiSys*, ACM, 2011.

[17] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies. The case for VM-Based cloudlets in mobile computing. *IEEE Pervasive Computing*, Vol. 8, pp.14–23, 2009.

[18] M. Shiraz *et. al.* A study on virtual machine deployment for application outsourcing in mobile cloud computing. *The Journal of Supercomputing*, Springer, Vol.63, pp.946–964, 2013.

[19] T. Verbelen, P. Simoens, F. D. Turck, and B. Dhoedt. Cloudlets: bringing the cloud to the mobile user. *Proc. of 3rd Workshop on Mobile Cloud Computing and Services*, ACM, 2012.

[20] Q. Xia, W. Liang, and W. Xu. Throughput maximization for online request admissions in mobile cloudlets. *Proc. of LCN*, IEEE, 2013.

[21] H. Yin, X. Zhang, T. Zhan, Y. Zhang, G. Min, and D. Wu. NetClust: a framework for scalable and pareto-optimal media server placement. *IEEE Trans. multimedia*, vol. 15, no. 8, pp. 2114–2124, 2013.

[22] Y. Zhang, H. Liu, L. Jiao, and X. Fu. To offload or not to offload: an efficient code partition algorithm for mobile cloud computing. *Proc. of CLOUDNET*, IEEE, 2012.