Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks

Mike Jia, Jiannong Cao, *Fellow, IEEE*, and Weifa Liang, *Senior Member, IEEE*

Abstract—Mobile applications are becoming increasingly computation-intensive, while the computing capability of portable mobile devices is limited. A powerful way to reduce the completion time of an application in a mobile device is to offload its tasks to nearby cloudlets, which consist of clusters of computers. Although there is a significant body of research in mobile cloudlet offloading technology, there has been very little attention paid to how cloudlets should be placed in a given network to optimize mobile application performance. In this paper we study cloudlet placement and mobile user allocation to the cloudlets in a wireless metropolitan area network (WMAN). We devise an algorithm for the problem, which enables the placement of the cloudlets at user dense regions of the WMAN, and assigns mobile users to the placed cloudlets while balancing their workload. We also conduct experiments through simulation. The simulation results indicate that the performance of the proposed algorithm is very promising.

Index Terms—Cloudlet placements, mobile user allocation, task response time minimization, mobile task offloading, mobile cloud computing

1 Introduction

Mobile applications are becoming increasingly computation-intensive, while the computing capacity of mobile devices are limited due to their portable sizes. A powerful approach to improving the performance of mobile applications is to offload some of their tasks to remote clouds, where an application consists of multiple tasks. Existing research in mobile task offloading mostly considered the cloud to be the remote offloading destination, due to its abundance of computational resources. However, the cloud usually is remotely located and far away from its users, and the network delay incurred by transferring data between users and the cloud can be very costly. This is especially undesirable in reality-augmenting applications and mobile multiplayer gaming systems, where a crisp response time is critical to the user's experience. Recent works [1], [2] have proposed the use of clusters of computers called cloudlets as a supplement to the cloud for offloading. Cloudlets are typically collocated at an access point (AP) in a network, and are accessible by users via wireless connection. A key advantage of cloudlets over the cloud, is that the close physical proximity between cloudlets and users enables shorter communication delays, thereby improving the user experience of interactive applications.

Despite the increasing momentum of cloudlet research in Mobile Computing, the question of where cloudlets should be placed within a network has largely been overlooked. Previous studies typically prescribed cloudlets for use in small private WLANs such as an apartment home, or an office floor. In such a setting, it can be argued that the cloudlet placement problem is trivial. Wherever the cloudlet is placed, the small size of the network means that the communication delay between the cloudlet and users is negligible. However, if we consider the use of cloudlets in Wireless Metropolitan Area Networks (WMANs), the problem of placement becomes much more significant.

Although there has been little research on the use of cloudlets in WMANs, we believe cloudlets are particularly suited for a WMAN environment. First, metropolitan areas have a high population density, meaning that cloudlets will be accessible to a large number of users. This improves the cost-effectiveness of cloudlets as they are less likely to be idle. Second, due to the size of the network, WMAN service providers can take advantage of economies of scale when offering cloudlet services through the WMAN, making cloudlet services more affordable to the general public. However, due to the size of the WMAN, a given user could be a significant number of hops away from the nearest cloudlet. Although the delay incurred per hop may be negligible when considering small networks, WMANs typically deal with much heavier traffic, resulting in lower quality of service and longer network delays. Because of this, long distances between the user and the cloudlet can adversely affect the performance of user applications, particularly those with a high data communication to processing ratio, such as online mobile games. At the same time, we must also carefully consider which users to assign to which cloudlets. A nearby cloudlet may have a short network latency to a user, but if the cloudlet is overloaded with other user task requests, the delay at the cloudlet could dominate the network latency. A better solution would be to route the user

- *M. Jia and W. Liang are with the Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia. E-mail: u5515287@anu.edu.au, wliang@cs.anu.edu.au.*
- *J. Cao is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk.*

Manuscript received 21 Nov. 2014; revised 22 May 2015; accepted 28 May 2015. Date of publication 25 June 2015; date of current version 6 Dec. 2017. Recommended for acceptance by S. Srirama.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2015.2449834

request to another cloudlet with a lighter workload. By strategically placing cloudlets in the WMAN and assigning user requests to cloudlets, we can minimize the total task request delay between users and cloudlets, bringing significant improvement to the performance of mobile applications.

The problem of cloudlet placement and user-to-cloudlet assignment in WMANs is a daunting one for a number of reasons. First, users in a WMAN are non-static and constantly moving around the city; the number of users in any particular area can vary over time. Deciding permanent cloudlet positions to best fit the dynamic movement and demands of the users poses a great challenge. Second, the scale of the problem itself is an enormous obstacle; the number of cloudlets that need to be placed in relation to a vast number of users makes traditional linear programming solutions infeasible. Assuming that all APs in the WMAN are viable cloudlet locations, the number of possible configurations is combinatorial; a heuristic solution is necessary to solve the problem in reasonable time. Finally, the assignment of users to cloudlets must be considered. We will later show that routing user requests to their "closest" cloudlets is not a satisfactory solution. Performing an optimal user-to-cloudlet assignment in conjunction with finding an optimal cloudlet placement, adds a new dimension of complexity to the already difficult problem.

In this paper we study the optimal cloudlet placement, and user-to-cloudlet assignment in a WMAN, to reduce the average wait time of offloaded tasks. We focus on solving the following optimization problem: given an integer number $K \geq 1$, place K cloudlets in the WMAN co-located with some APs and assign the users to the cloudlets such that the average wait time for offloaded user requests is minimized.

In this paper we make the following contributions. We first devise a novel multi-user, multi-cloudlet, system model using queuing network. We then propose an algorithm for the K cloudlet placement and user to cloudlet assignment problem. We also discuss how our algorithm can be practically applied to WMANs with dynamic and constantly moving user. We finally perform simulation experiments which indicate that our algorithm delivers a solution with near optimal performance according to simulation results.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 introduces the system model and the problem definition. Section 4 gives a detailed description of the proposed cloudlet placement algorithms, and Section 5 conducts a case study and simulation results. A conclusion is drawn in Section 6.

2 Related Work

Mobile task offloading to cloudlets is a new and important research topic. Cloudlets are typically described as resource-rich computers deployed at APs in a network, and act as offloading destinations of mobile users [1], [2], [3]. As cloud servers are usually physically far away from their users, communication delay between each user and the remote cloud can be long and unpredictable [1]. This can be especially problematic for mobile applications where a crisp response time is critical to the user experience, such as reality-augmenting applications and mobile gaming systems. The close physical proximity between users and cloudlets means that the delay involved in task offloading is greatly reduced when compared to the remote cloud, thereby improving user experiences. To offload tasks, mobile users encapsulate tasks in a virtual machine (VM) [4] which is then uploaded to the cloudlet for execution. The user can then offload more tasks using offloading operations on its VM in the cloudlet. Once a task on the VM in the cloudlet has been executed, the result is returned back to the user.

Although existing frameworks and algorithms for task offloading mostly focused on clouds [5], [6], cloudlets have been quickly gaining recognition as an alternative offloading destination [7], [8], [9], [10], [11]. Odessa [10] is an example that can offload tasks to either the cloud or a dedicated cloudlet. The study in [12] further considered the situation where a user can offload his/her tasks to both the cloud and the cloudlet at the same time, and proposed a solution based on game theory. Cloudlets have also impacted the research of mobile cloud gaming [13], [14]. For example, a cloudlet-assisted multiplayer cloud gaming system was proposed in [14], which uses cloudlets as caches for video frames.

Several recent papers further broadened the definition of cloudlets to include ad-hoc computers in the network. In [16], [17], Verbelen et al. proposed such a cloudlet architecture, creating a framework which enables ad-hoc discovery of nearby devices in the network to share resources. While such an architecture does have its benefits, there are serious security and privacy concerns when offloading tasks to ad-hoc computers, so for this paper we restrict our definition of cloudlets to being a cluster of computers co-located with an AP.

In this paper we consider the cloudlet placement problem in a WMAN. A WMAN is a computer network that provides wireless Internet coverage for mobile users in a metropolitan area, and are often owned and operated as public utilities [18]. Intuitively, the cloudlet placement problem is similar to the facility location problem [19], they however are essentially different. Each user can be self-sufficient without a cloudlet as it is able to offload tasks to the remote cloud. This makes it difficult to effectively apply traditional facility location algorithms [20] to our problem. To the extent of our knowledge, this is the first piece of work to consider the placement of cloudlets in WMANs.

3 Preliminaries

In this section we first describe the WMAN system model, and the task offloading protocols for users and cloudlets in our system model. We then define the problem precisely.

3.1 WMAN System Model

A WMAN can be represented by a set of access points $P = \{p_1, \ldots, p_m\}$ interconnected by the Internet, and a set of mobile users $U = \{u_1, \ldots, u_n\}$ who can access the network through APs. We use an undirected graph $G(V, E)$ to represent the connections between users and APs in a WMAN, where $V = P \cup U$. There are two types of edges in G. An edge between a user u_i and an AP p_j $(u_i, p_j) \in E$, indicates that u_i is wirelessly connected to p_j (illustrated by the dotted lines in Fig. 1). An edge between two APs p_i and p_j

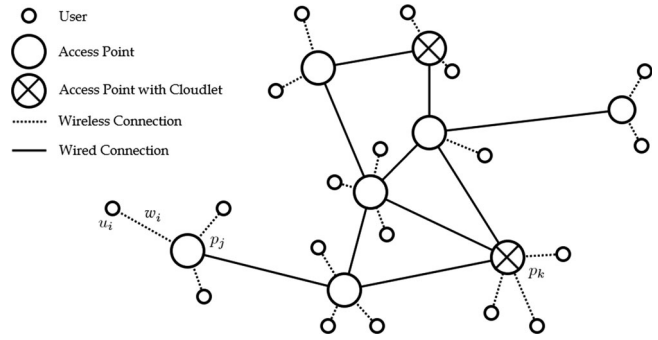


Fig. 1. A WMAN example.

$(p_i, p_j) \in E$, indicates that they have a one-hop distance (illustrated by the solid lines in Fig. 1. We assume that graph G is connected, which implies that all APs are reachable with each other through high speed Internet connection. In addition, there is also a remote cloud that can be accessed from every AP via the Internet.

The tasks of each mobile user can fluctuate unpredictably, especially when the user is running multiple applications at the same time. We assume that each user u_i has a stream of offloadable tasks that randomly arrive into the system with arrival rate λ_i, according to the Poisson process. The core functions of the application that cannot be offloaded continue running on the device, but these tasks are invisible to our system model.

To offload its tasks to cloudlets for processing, a user needs to relay its offloaded tasks through the network G. Taking Fig. 1 as an example, let w_i denote the wireless delay between u_i and the AP p_j that u_i is connected. If the offloaded task of u_i will be processed in the cloudlet located at AP p_k, then the task needs to be relayed from p_j to p_k. We assume that all offloaded tasks are of a uniform data packet size, and so delays incurred in transferring any task through the network between a pair of APs is identical. To model such a network delay in the WMAN, denote by $D \in \mathbb{R}^{m \times m}$ the network delay matrix, where entry $D_{j,k}$ represents the communication delay in relaying a task between AP p_j and the AP p_k.

3.2 Offloading System Model

In this subsection we introduce a system model for multi-user mobile task offloading. The system is modeled as a queuing network. We assume that there are K cloudlets to be placed in G. Offloaded tasks can either be executed on one of the K cloudlets or the remote cloud. Each user u_i begins by offloading its task stream to the cloudlet with task arrival rate λ_i. If the cloudlet is overloaded at this moment, it will offload a fraction of its arriving task stream to the remote cloud.

Cloudlets are modeled as a $M/M/c$ queue, where every cloudlet consists of c homogeneous servers with fixed service rate μ. The wait time of a task request arriving at a cloudlet consists of a queue time and a service time. We define a function f_Q which takes a given task arrival rate (workload) λ, and returns the average queue time

$$f_Q(\lambda) = \frac{C\left(c, \frac{\lambda}{\mu}\right)}{c\mu - \lambda}, \tag{1}$$

where

$$C(c, \rho) = \frac{\left(\frac{(c\rho)^c}{c!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{c-1} \frac{(c\rho)^k}{k!} + \left(\frac{(c\rho)^c}{c!}\right)\left(\frac{1}{1-\rho}\right)}. \tag{2}$$

Formula (2) is known as Erlang's formula (2) [21].

Let U_j denote the set of users assigned to the cloudlet placed at p_j, where $U_j = \{u_i \mid y_{i,j} = 1\}$. Due to the limited computing capacity of the cloudlet, it may not be possible for a cloudlet to serve all the user task requests it received. If the workload of a cloudlet is too heavy, the queue time can become excessively long, potentially slowing down the mobile user's application. As a result it is not uncommon for cloudlets to offload tasks to the remote cloud when overloaded [1], [12]. Thus, we assume that the maximum workload at each cloudlet is capped at an arrival rate of λ_{max}, and that the remaining task requests will be offloaded to the remote cloud. We determine the fraction ϕ_j of tasks the cloudlet processes as follows.

$$\phi_j = \begin{cases} 1 & \text{if } \lambda_{max} > \lambda(j) \\ \frac{\lambda_{max}}{\lambda(j)} & \text{otherwise,} \end{cases} \tag{3}$$

where U_j denotes the set of users assigned to the cloudlet at AP p_j, and $\lambda(j) = \sum_{u_i \in U_j} \lambda_i$. The wait time of each task in the cloudlet at p_j is

$$t_{clt}(j) = f_Q\left(\phi_j \cdot \sum_{u_i \in U_j} \lambda_i\right) + 1/\mu. \tag{4}$$

Offloaded tasks will be relayed to the remote cloud through the Internet, we thus assume the transfer of the task through the Internet incurs a fixed delay B. We further assume that the cloud has sufficient computing resource to process the tasks, and the queue time of the tasks in the remote cloud is negligible. We model the cloud as a $M/M/\infty$ queue with the service rate μ (the same as all cloudlets). The wait time t_{cld} of offloaded tasks at the remote cloud can be calculated as follows:

$$t_{cld} = B + 1/\mu \tag{5}$$

From Formulas (3), (4), and (5), the average wait time of offloaded tasks by user u_i is calculated as follows:

$$t_i = w_i + D_{k,j} + \phi_j \cdot t_{clt}(j) + \left(1 - \phi_j\right) \cdot t_{cld}, \tag{6}$$

where u_i is wirelessly connected to AP p_k, and assigned to the cloudlet located at p_j.

The average wait time of offloaded tasks by all users in the system in Eq. (6) is referred to as the *system response time (SRT)*

$$t = \frac{1}{n}\sum_{i=1}^{n} t_i. \tag{7}$$

All symbols used thus far are listed in Table 1.

TABLE 1
System Symbols

Symbol	*Definition*
G	Network graph
X	Vector position of cloudlets at APs
Y	Matrix assignment of users to cloudlets
n	Number of users
m	Number of APs
c	Number of servers per cloudlet
λ_i	Task arrival rate/workload for user u_i
$\lambda(j)$	Task arrival rate/workload at cloudlet p_j
w_i	Wireless delay between user u_i and its AP
$D_{i,j}$	Network delay between AP p_i and p_j
$f_Q(\lambda)$	Function for cloudlet queue time. takes workload λ as input
T_{net}	Tolerable network delay threshold
λ_{max}	Maximum cloudlet workload
ϕ_j	Fraction of tasks accepted at cloudlet p_j
μ	Cloudlet/cloud server service rate
B	Internet delay
$t_{clt}(j)$	Cloudlet wait time at p_j
t_{cld}	Remote cloud wait time
t_i	Response time of user u_i
t	System response time

3.3 Problem Formulation

We introduce two sets of variables X and Y to represent the placement of cloudlets at APs and the assignment of users to cloudlets, where x_j indicates whether or not a cloudlet will be placed at AP p_j, and $x_j = 1$ means that a cloudlet is placed at p_j, otherwise, $x_j = 0$. In Fig. 1, two cloudlet are placed at AP p_j and p_k, as a result, $x_j = x_k = 1$ while all other values of x are zeros. Y represents the assignment of users to cloudlets, $y_{i,j} = 1$ indicates that user u_i is assigned to the cloudlet at AP p_j; otherwise $y_{i,j} = 0$. We assume that all cloudlets will be co-located with some of the AP locations,

$$X = \{\, x_j \mid 1 \le j \le m \,\},$$

and

$$Y = \{\, y_{i,j} \mid 1 \le i \le n,\, 1 \le j \le m \,\}.$$

We also introduce a tolerable network delay threshold T_{net} for network delay between users and cloudlets. While it may not be possible to provide a solution such that every user has a network delay below the threshold T_{net}, the introduction of T_{net} will have practical meanings when we devise an algorithm that can deliver a feasible solution. In the following we define parameters related to the problem. Let Λ be the set of task arrival rate of users,

$$\Lambda = \{\lambda_i \mid 1 \le i \le n\}, \tag{8}$$

and let W be the set of wireless delays between users and their APs,

$$W = \{w_i \mid 1 \le i \le n\}. \tag{9}$$

Recall network delay matrix D, the maximum cloudlet load λ_{max}, the Internet service rate B, the cloudlet/cloud server service rate μ, the number of servers c in each cloudlet, and the tolerable network delay threshold T_{net}.

The K cloudlet Placement Problem (KCP) in a WMAN G is defined as follows. Given an integer $K \ge 1$ and system model parameters $(G, \Lambda, W, D, T_{net}, \lambda_{max}, B, \mu, c)$, the problem is to find X (the placement of cloudlets among the APs) and Y (the assignment of users to the cloudlets) such that the system response time t in Eq. (7) is minimized, i.e.,

$$\min_{X,Y} t. \tag{10}$$

4 Cloudlet Placement to Minimize Response Time

In this section we propose two heuristic algorithms for the K cloudlet placement problem. We start with a simple Heaviest-AP First (HAF) algorithm. We then propose a Density-Based Clustering (DBC) algorithm that surmounts the shortcomings of the HAF algorithm.

4.1 Heaviest-AP First Placement

We first find a cloudlet placement X in the WMAN. As the objective of the problem is to reduce the system response time by bringing cloudlets closer to users, a plausible approach is to place cloudlets at the APs where the user workloads are the heaviest. We thus sort the APs in the network by the accumulative task arrival rate of users, and place cloudlets at the top K APs with the largest workloads. We then assign users to the cloudlets as follows.

For each user u_i connected to AP p_k, we find the cloudlet to which AP p_k has the smallest network delay $D_{k,j}$, and assign u_i to that cloudlet. This minimizes the network delay between a user and his/her cloudlet. Algorithm 1 gives the details of the Heaviest-AP First Placement.

Algorithm 1. *Heaviest-AP First Algorithm*

Input: $(K, G, \Lambda, W, D, T_{net}, \lambda_{max}, B, \mu, c)$
Output: (X, Y).

```
/* Cloudlet placement */
Q ← ∅ /* Q is the set of cloudlet locations */ ;
for k ← 1 to K do
    j ← k, where k is an index of an AP p_k with
    Σ_{u_i ∈ user(p_k)} λ_i = max_{p'_k ∈ V−Q} {Σ_{u_i ∈ user(p'_k)} λ_i}, and
    user(p_k) is the set of users connected to AP p_k;
    Q ← Q ∪ {p_j};
    X[j] ← 1;
/* User-to-cloudlet assignment */
for i ← 1 to n do
    let p_{k'} be the AP that user u_i) is connected;
    find a cloudlet j and assign the tasks of u_i to it,
    where D_{k,j} = min_{p_j ∈ Q} {D_{k,j}};
    Y[i, j] ← 1.
```

There are two major shortcomings of the HAF algorithm. First, the APs with the heaviest workload are not always the closest ones to their users. Consider the WMAN in Fig. 2, although AP p_j may have the heaviest user workload, the users wirelessly connected to p_j are all outliers in the network. As most users in the network are multiple AP hops

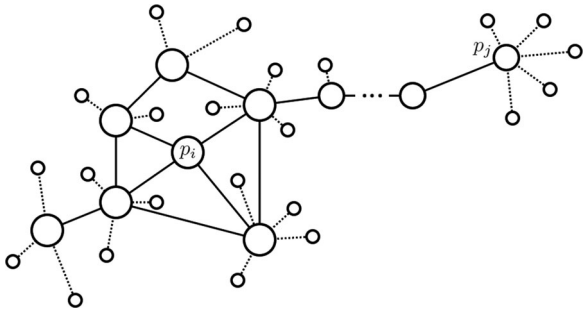


Fig. 2. User dense regions in a WMAN.

away from p_j, it is clear that the location of p_j is not the best candidate for cloudlet placement. On the other hand, even though there are no users connected to AP p_i, the majority of users in the WMAN are within a single hop of p_i, thus making p_i a more favorable cloudlet location than that of p_j. As a result, it is important that we target APs positioned in user-dense regions of the WMAN for cloudlet placement, instead of APs with the heaviest user workload.

The second issue with the HAF algorithm is that assigning users to their closest cloudlets can lead to uneven distributions of users to cloudlets. This can cause certain cloudlets to be overloaded while others are under-utilized. Fig. 3 shows the relationship between the average wait time of offloaded tasks to a cloudlet and the number of users using the cloudlet. With the increase on the number of users, the wait time of tasks sharply increases. Once the cloudlet reaches its maximum workload λ_{max}, as seen in Fig. 3, the cloudlet begins to offload its surplus user requests to the remote cloud. This implies that a better strategy is to assign some user tasks to those under-loaded cloudlets rather keeping them on an overloaded cloudlet. Therefore, it is clear that workload balancing among the cloudlets is an important issue for user assignment to cloudlets.

4.2 Density-Based Clustering Placement

To surmount the shortcomings of the HAF algorithm, we now introduce our main solution to the K cloudlet placement problem. Through the discussion on the HAF algorithm, we have two key observations that will guide us to develop an efficient solution to the problem. First, we target user-dense regions of the WMAN for cloudlet placement. This means that cloudlets are situated close to large numbers of users, thereby reducing the average network latency between users and cloudlets. Second, we balance user workloads among the cloudlets, which can dramatically reduce the average cloudlet queue time of tasks. We propose a density based approach to cloudlet placement. Recall that the given parameter T_{net} is the tolerable network delay threshold. Let $U_{T_{net}}(j)$ be the set of users that can access AP p_j with the network delay time no greater than T_{net}, where

$$U_{T_{net}}(j) = \{\, u_i \mid D_{k,j} \leq T_{net} \,\},$$

where p_k is the AP to which u_i is wirelessly connected. We call $U_{T_{net}}(j)$ the set of candidate users at AP p_j. As the candidate users of p_j have short network delays to it, the candidate users are clustered around the AP. It can be argued that a larger number of users in the set of candidate users of

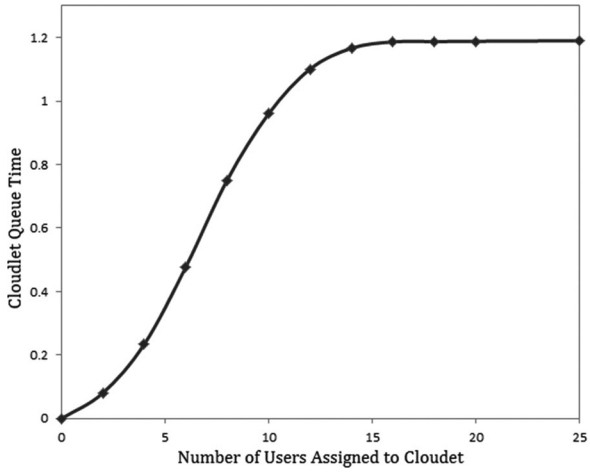


Fig. 3. Cloudlet queue time.

an AP, indicates a higher user-density in the region of the AP. Let $\lambda_{T_{net}}(j)$ denote *the candidate workload* of AP p_j, then

$$\lambda_{T_{net}}(j) = \sum_{u_i \in U_{T_{net}}(j)} \lambda_i.$$

To place the cloudlets to APs, we first choose an AP p_j with the largest candidate workload $\lambda_{T_{net}}(j)$, and place a cloudlet at p_j. We then remove the set of users directly connected to p_j from network G, recalculate each AP's candidate users in the updated network, and find the next AP with the largest candidate workload. We repeat this process K times until all K cloudlets are placed.

As APs close to each other will often share candidate users, an AP is very likely to have a high number of candidate users if its neighboring APs also have high numbers of candidate users. As a result, regions of the WMAN with a higher user-density will typically have more APs chosen for cloudlet placement. Furthermore, by removing the users of AP p_j after a cloudlet is placed at the AP, we can reduce the likelihood of oversaturating a densely populated network region with cloudlets. This means that the distribution of cloudlets in the WMAN roughly follows the distribution of mobile users, thus making it easier to balance the workload among cloudlets when assigning them with users.

We then solve the user-to-cloudlet assignment problem as follows. We find the cloudlet with the heaviest candidate user workload. We denote λ_{avg} to be the average workload among the cloudlets, where

$$\lambda_{avg} = \frac{1}{K} \cdot \sum_{i=1}^{n} \lambda_i.$$

We then assign the cloudlet's candidate users to itself, until the workload of the cloudlet exceeds λ_{avg}. Once a set of users have been assigned to the cloudlet, we remove the assigned users and the AP of the cloudlet from the WMAN graph G. We then move to the next cloudlet with the largest number of candidate users and assign its candidate users to itself in the same way. This process is repeated until all cloudlets have been given the opportunity to assign their candidate users. Finally, the remaining unassigned users in G will be assigned to their closest cloudlets.

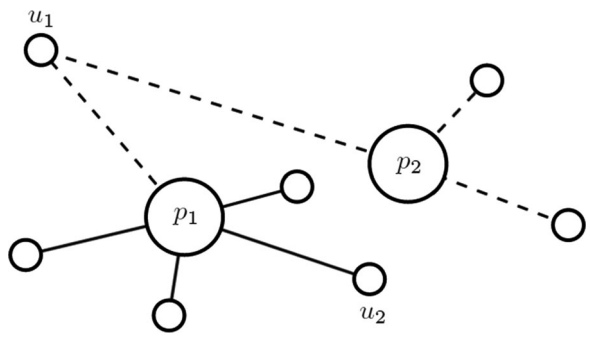


Fig. 4. Cloudlets at p_j and $p_{j'}$.

In this algorithm, the last issue to be addressed is how to prioritize candidate users for assignment to the cloudlets.

4.2.1 Relative Distance User-Assignment to Cloudlets

Because each cloudlet can only serve a limited number of users, it is important to decide which candidate users to assign to which cloudlets. A naive solution is to connect the closest candidate users to the cloudlet, until the cloudlet exceeds the average cloudlet workload. However this can lead to a situation where a user could be forced to connect to a far away cloudlet. Fig. 4 shows such an example, where there are two cloudlets, placed at p_1 and p_2. Suppose we assign users to the cloudlet at p_1 and that the cloudlet will achieve the average workload after assigning four users. If we assign the four closest users, then user u_1 is forced to connect to an alternative cloudlet p_2, which is significantly further away from the user than p_1. The problem is that while users like u_2 are closer to p_1 than u_1, they also have a close alternative cloudlet that they can connect to, whereas u_1's alternative cloudlet is relatively far away. A better solution would be to assign u_1 to p_1, and u_2 to p_2. This is because the relative distance of u_1 to p_1 against p_2 is closer than u_2's relative distance to p_1 against p_2 (thus u_1 should have a higher priority than u_2 to connect to p_1).

It is clear that some users should have a higher priority when being assigned to a cloudlet, based on their distances to the cloudlet relative to the distances of an alternative cloudlet. Now we introduce the concept of "relative distance" between a user and a cloudlet. Suppose u_i is a candidate user of p_j, and that $p_{j'}$ is the closest cloudlet to u_i that is not p_j (we call $p_{j'}$ the alternative cloudlet of u_i). Let $r_{i,j}$ denote the relative distance of u_i to the cloudlet at p_j, where u_i is wirelessly connected to AP p_k,

$$r_{i,j} = \frac{w_i + D_{k,j}}{w_i + D_{k,j'}}.$$

Here we use delay to infer the distance between the user and the cloudlets. Note that w_i denotes the wireless delay between u_i and its AP p_k, while $D_{k,j}$ denotes the network delay between AP p_k and the cloudlet at p_j.

When the relative distance between u_i and p_j is low, this means that u_i's alternative cloudlet is very far away. In the extreme case, if the relative distance is zero, this indicates that there is no alternative cloudlet available for it. As a result, u_i should be given a higher priority when assigning users to p_j, than a user with a greater relative distance. Taking Fig. 4 as an example, u_1 has relative distance to p_1 that is less than one, as it is closer to p_1 than p_2. u_2 on the other hand, has a relative distance to p_1 that is greater than one as it is further away from p_1 than p_2. Therefore, u_1 should be given a higher assignment priority than u_2 when being assigned to p_1, as it has a smaller relative distance to p_1 than u_2. As relative distance increases, the user has less priority, because it is closer to its alternative cloudlet.

To assign users to p_j, we first compute the relative distance to p_j for each user u_i. We then connect the user by increasing order of $r_{i,j}$ until the average cloudlet workload is exceeded. We finally remove p_j and the set of assigned users from further consideration. Algorithm 2 gives the detailed description of our method.

Algorithm 2. *Density-Based Clustering Algorithm*

Input: $(K, G, \Lambda, W, D, T_{net}, \lambda_{max}, B, \mu, c)$;
Output: (X, Y).

```
/* Cloudlet placement */
U' ← U /* the set of unassigned users for cloudlets */;
Q ← ∅ /* Q is the set of cloudlet locations */;
for iteration ← 1 to K do
    find a cloudlet j such that λ_{T_net}(j) = max_{p_k∈V}{λ_{T_net}(k)};
    Q ← Q ∪ {p_j};
    U' ← U' − user(p_j) where user(p_j) is the set of users
    connected to AP p_j;
    X[j] ← 1;
/* User-to-cloudlet assignment */
U' ← U;
for k ← 1 to K do
    We find a cloudlet p_j such that
    |U_{T_net}(j)| = max_{p_{j'}∈Q}{|U_{T_net}(j')|};
    let R be the list of elements in U_{T_net}(j) ordered by r_{i,j},
    where r_{i,j} is the relative distance between user u_i
    and cloudlet p_j;
    for l ← 1 to |U'| do
        Y[R[l], j] ← 1;
        if r_{R[l],j} ≠ 0 then
            if λ(j) > λ_avg then
                /* λ(j) is the workload at cloudlet p_j */
                exit;
        Q ← Q − {p_j};
        U' ← U' − {u_i | u_i ∈ U_j} /* remove assigned users
        from U' to prevent them from being reassigned */.
```

4.3 Approach for Dynamic WMANs

As mentioned, the main challenge of cloudlet placement in a WMAN is the dynamic nature of its user movement. Users are constantly moving around within the network, and the network delay fluctuates throughout the day. So far we have described how to solve the K cloudlet placement problem in a static snapshot of the WMAN, but most WMANs cannot be accurately represented by a single snapshot. However, by using historical data of its users to construct multiple snapshots of a WMAN, we solve the KCP problem under this dynamic setting, by extending the solution in the previous section through necessary modifications. In the following we will give a baseline method of how the historical data of user movements can be used in the cloudlet placements and user allocations to the placed cloudlets.

4.3.1 Cloudlet Placement in WMANs

Mobile user movement in a WMAN often follows repetitive patterns. Users typically stay at home in the suburbs at night, and commute to work in the city-center during the day. Consider a cloudlet service provider who has constructed N snapshots of the WMAN at regular time intervals already based on the historical data. Let $G_i(V_i, E_i)$ denote the snapshot of the WMAN at the ith snapshot time interval. We solve X_i and Y_i for the KCP problem with system parameters $(G_i, \Lambda_i, W_i, D_i, T_{net}, \lambda_{max}, B, \mu, c)$ for G_i for each snapshot time interval i with $1 \leq i \leq N$. Denote by $\overline{X}$ the collection of all cloudlet locations from the snapshot solution X_i for each i with $1 \leq i \leq N$,

$$\overline{X} = \{p_j \mid \exists i \text{ s.t. } X_i(j) = 1\}.$$

As user workloads change overtime, we can expect that the placement of cloudlets X_i at each snapshot time interval i is different. In order to find a cloudlet placement to better suit for all the N snapshots, the cloudlet placement will be slightly further away from the users in a given snapshot i when compared to its snapshot-level cloudlet placement X_i on average. Our objective thus is to choose a cloudlet placement $M = \{m_1, m_2, \ldots, m_K\}$ to minimize the additional network delay, when applying M to snapshot i, compared to the snapshot-level cloudlet placement X_i. Because users often stay for extended periods of time in certain areas, such as offices, coffee houses, shopping centers, the set of all snapshot cloudlet placements $\overline{X}$ will tend to exhibit cloudlet placement locations that are clustered together. We can achieve our objective, by exploiting this property, through applying the K-medians clustering algorithm to the set of all snapshot cloudlet placements $\overline{X}$.

A K-medians clustering algorithm is a variation of the K-means algorithm, which iteratively finds K clusters $\{C_1, C_2, \ldots, C_K\}$ in a set of locations, each with a median location at its center $\{m_1, m_2, \ldots, m_K\}$, to minimize the within-cluster sum of squares [22]. We define the median location m of a cluster C to be the location in the cluster that has the smallest sum distance from every other location in the cluster, $m = p_k$ such that $\sum_{p_j \in C} D_{k,j} = \min_{p_{k'} \in C} \{\sum_{p_j \in C} D_{k',j}\}$.

If we apply the K-median algorithm on the set of cloudlet locations $\overline{X}$, we can find the global cloudlet placement $M = \{m_1, m_2, \ldots, m_K\}$ that minimizes within-cluster network delay,

$$\arg \min_{\{C_1,\ldots,C_K\}} \sum_{i=1}^{K} \sum_{p_j \in C_i} D_{j,m_i},$$

where $m_1, m_2, \ldots, m_K$ are the medians of the clusters $C_1, C_2, \ldots, C_K$, respectively. Due to the overlapping of cloudlet placements across different snapshots, the clusters $C_1, C_2, \ldots, C_K$ represent the general regions of cloudlet locations across all snapshots. Thus, by minimizing within-cluster network delay, we also minimize the additional network delay between the global cloudlet placement and the snapshot-level cloudlet placements. Algorithm 3 provides the details of the K-medians clustering algorithm.

Algorithm 3. *K-medians clustering algorithm*

Input: $(K, G, \Lambda, W, D, T_{net}, \lambda_{max}, B, \mu, c, \overline{X})$,
Output: M.

```
1:  M[] ← getRandomK(X̄) /* Initialize medians to random
    locations in X̄ */;
2:  /* Let MAX be the maximum number of iterations */;
3:  for iteration ← 1 to MAX do
4:    for j ← 1 to |X̄| do
5:      /* Assign each location p_j in X̄ to the cluster of the
        closest median. */;
6:      i ← k such that D_{M[k],j} = min_{k'} {D_{M[k'],j}};
7:      C_i ← C_i ∪ {p_j} /* cluster C_i */;
8:    for k ← 1 to K do
9:      m ← p_j such that Σ_{p_i∈C_k} D_{i,j} = min_{p_j'∈C_k} {Σ_{p_j'∈C_k} D_{i,j'}}
        /* Re-calculate median m_k in cluster C_k. */;
10:     M'[k] ← m;
11:   if M = M' then
12:     /* If cluster formations have converged */
13:     exit;
14:   else
15:     M ← M'.
```

4.3.2 AP-to-Cloudlet Assignment in WMANs

To address user-to-cloudlet assignment in a dynamic WMAN, we are aware of the fact that a WMAN in the real world can potentially have millions of users, and such large-scale makes it infeasible to individually assign users to cloudlets. Furthermore, the assignment needs to be constantly updated as users move around within the network. One way we can address this issue is to group users connected to the same AP together, and then collectively assign them to one cloudlet. More formally, given a current snapshot of the WMAN $G(P \cup U, E)$, a network delay matrix D, and the current cloudlet placement configuration X, we create a new graph $G_P(P \cup U_P, E_P)$, where each AP $p_j \in P$ has a single user u_j. We then define the set of task arrival rates for the new users $\Lambda_P = \{\lambda(j) \mid 1 \leq j \leq m\}$, where $\lambda(j) = \sum_{u_i \in U_j} \lambda_i$. As we have aggregated all users into their connected APs, we set the wireless delay w_i for each new user to zero. We then solve the user-to-cloudlet assignment problem with system parameters $(G_P, \Lambda_P, D, T_{net}, \lambda_{max}, B, \mu, c, X)$. By reducing the number of users to APs, we can significantly decrease the computational complexity of the problem, allowing the algorithm to be run in real time. Furthermore, assigning APs to cloudlets allows users to move freely in the network, without being tied down to a specific cloudlet. When a user moves across the network and connects to a different AP, the user "discovers" his/her newly assigned cloudlet via AP-to-Cloudlet assignment.

5 Simulation

In this section we evaluate the performance of the proposed algorithms in simulation environments. We begin by explaining the simulation environment setting. We then apply the proposed algorithms to a wireless network in Hong Kong as a

TABLE 2
System Parameters

Symbol	*Definition*	*Default Value*
n	Number of users	150
m	Number of APs	50
K	Number of cloudlets	5
l	Attachment rate of the network	1.5
λ_i	Task arrival rate/workload for user u_i	$0 \leq \lambda_i \leq 2.99$
w_i	Wireless data-rate between user u_i and its AP	$0.1 \leq w_i \leq 0.4$
μ	Cloudlet/cloud server service rate	10
c	Number of servers in each cloudlet	5
B	Internet delay	0.8
λ_{max}	Maximum cloudlet workload	45

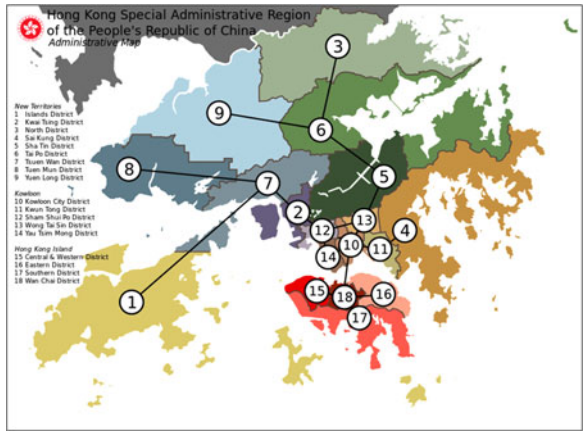


Fig. 5. Hong Kong MTR map.

case study. We finally evaluate the performance of the proposed algorithms in randomly generated networks.

5.1 Simulation Environment

In our system model, we define a number of parameters for the K cloudlet placement problem. Each system parameter has a default value which it is set to, unless the parameter is the free variable in the experiment.

In each trial, the task arrival rate of user u_i, or workload λ_i, is determined by the Normal distribution with an average of 2, and a variance of 0.5. λ_i is capped at 2.99. w_i is also randomly generated according to the Normal distribution, with an average of 0.2, variance of 0.1, and $0.1 \leq w_i \leq 0.4$. The service rate for cloud and cloudlet servers is 10, where each cloudlet has five servers. The Internet delay B is 0.8 which can be expected if the remote cloud is in another country, and finally the maximum cloudlet workload λ_{max} is 45.

To construct the network delay matrix $D_{i,j}$, we compute the distance between each pair of APs in network G in terms of number of hops, using Dijkstra's algorithm. Table 2 contains the default values for the system parameters in our simulation.

5.1.1 Random Network Generation

As part of our evaluation, we tested the proposed algorithms on randomly generated networks. Public data for wireless network topologies (especially on the scale of WMANs) is difficult to obtain, so some degree of inference is required when modelling WMAN topologies. The topology of the Internet is well known to be a scale free network (i.e., it follows a power-law degree distribution) [23]. As a result, it is reasonable to suspect that the network on the router-level is likewise scale-free. We therefore assume that a WMAN can be modelled as a scale-free network. In our simulation, we use the Barabasi-Albert Model [23] to generate random scale-free networks. A Barabasi-Albert network generator has two parameters: the number of nodes (APs) m, and the attachment rate of the network l. Beginning with an initial connected graph of l nodes, we add a new node to the network one at a time. Each new node added randomly "attaches" to a maximum of l nodes in network, forming a maximum of l new links. The probability of the new node connecting to node i, P_i, is determined by the following formula:

$$P_i = \frac{k_i}{\sum_{j \in V} k_j},$$

where k_i is the degree of node i at this moment, and V is the set of existing nodes in the network. As a result of the probability function, nodes with higher degree are more likely to connect to the new node, thereby encouraging the formation of clusters. The network generator continues in this manner until there are m nodes in the network. Each link in the network is assigned a network delay time: $0.1 \leq \mathcal{N}(0.15, 0.05) \leq 0.2$, where $\mathcal{N}(0.15, 0.05)$ is a randomly generated number between 0.1 and 0.2, according to the normal distribution. This randomizes the delay in the network while preserving triangle distance inequality, as any pair of nodes with 2 degrees of separation has an intermediary distance of at least 0.2. Finally, we randomly assign each of the n users to an AP in the network, according to the uniform distribution.

5.2 Case Study

Hong Kong is a densely populated city with public WiFi access in every subway station, as well as most public areas [24]. In this case study, we imagine that Hong Kong's main WiFi service provider HKBN, is planning to provide cloudlet services to users in Hong Kong. The number of potential cloudlet locations are limited to the 18 districts in Hong Kong, each of which has a small network hub interconnected, with other district hubs via wired connection. Fig. 5 shows a map of Hong Kong's districts which we will use as a template for our WMAN. Although the network topology is not publicly available, we can infer the wired connections between each district hub, using the Hong Kong Mass Transit Railway map to represent wired hub-to-hub edges in the WMAN.

It is reasonable to assume that the network topology has a close resemblance to the public transportation infrastructure. This is mainly due to two reasons: first, public transport stations are often densely populated, and thus are good candidate locations to set up APs and cloudlets. Second, while it is common for connections between network APs and switches to be wireless, it is often practical to have wired connections between certain APs, such as those

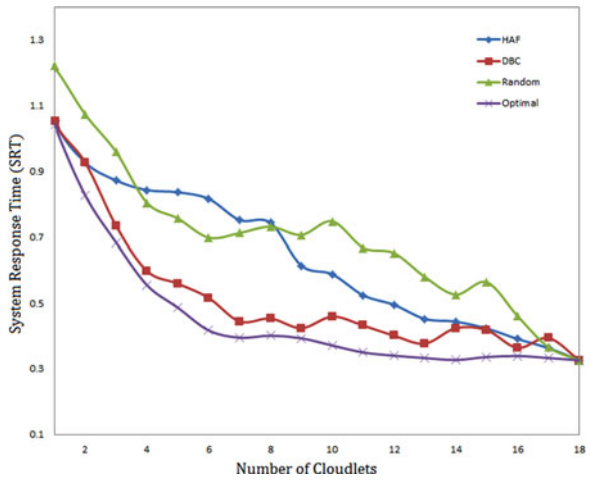


Fig. 6. System response times achieved by cloudlet placement algorithms.

linking network "hubs" for higher bandwidth and reliability. Having the broadband cables follow transportation routes is practical and sometimes necessary, for example, the Cross-Harbor Tunnel hosts Internet cable connections between Hong Kong Island and the mainland.

Using publicly available information, we can estimate task arrival rates of APs Λ_P based on the user population in local districts. We then solve the KCP problem with system parameters $(G_P, \Lambda_P, D, T_{net}, \lambda_{max}, B, \mu, c)$.

5.2.1 Performance Evaluation of Cloudlet Placement Algorithms

To evaluate the performance of the proposed Heaviest-AP-First and Density-Based-Clustering algorithms, we use two benchmark algorithms: one is a random cloudlet placement that randomly places cloudlets at K AP locations in the network, and assigns users to their closest cloudlets. Simulation results of the random cloudlet placement algorithm are the average across 100 trial runs. Another benchmark is the optimal algorithm, which iterates through all the possible cloudlet placements and user-to-cloudlet assignments to find the optimal system response time. As the WMAN in our case study has a very abstract form with only a limited number of users and APs, we are able to exhaustively search for the optimal performance.

In Fig. 6 we study how SRT responds with the increase of the number of cloudlets in the network. The DBC and optimal algorithms decrease in their SRTs from $K = 1$ to $K = 18$, before eventually plateauing. When $K = 1$, the majority of offloaded tasks are further offloaded to the remote cloud. Adding a single cloudlet decreases the SRT by 21 percent, and continues to linearly decrease until $K = 6$. However after $K = 6$, there is a sufficient number of cloudlets placed within the network such that each cloudlet is able to accept all task requests. As users are able to offload all their tasks to the cloudlets they assigned, SRT is bounded from below by wireless and network latency between users and cloudlets, causing SRT to plateau. This suggests that after a certain number of cloudlets have been placed, the placement of additional cloudlets will follow the law of diminishing returns in terms of SRT.

The DBC algorithm delivers a near optimal performance, with its SRT of only 11 percent greater than that of the optimal one on average. The HAF algorithm, on the other hand, has a lackluster performance with an average SRT of 34 percent greater than the optimal one, while the random placement, trails even further behind the proposed algorithms, with an average SRT of 46 percent larger than the optimal one. The performance of the HAF algorithm is quite poor when K is small, in between the range $3 \leq K \leq 7$, the SRT achieved by the HAF placement is even worse than that by a random placement. To gain a better understanding of why this is the case, we visually compare the cloudlet placements produced by the proposed algorithms when $K = 5$.

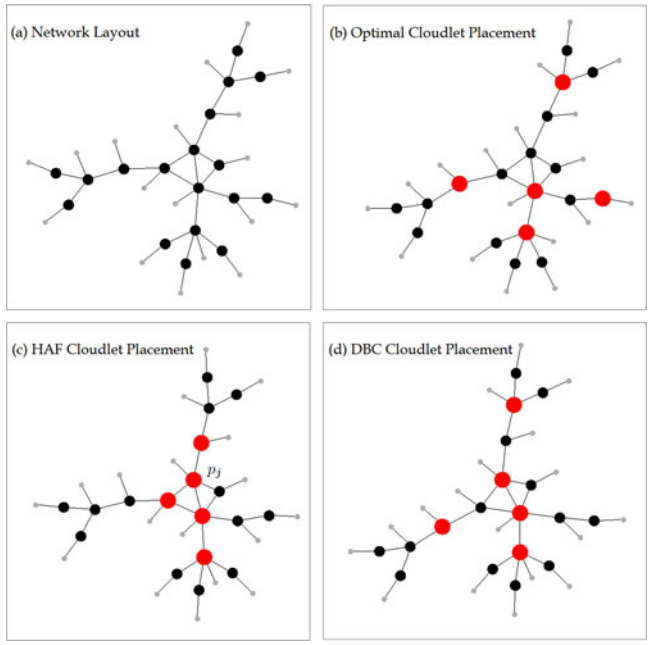


Fig. 7. Visual comparison of cloudlet placement positions for each placement algorithm when $K = 5$.

It can be seen from Fig. 7c that the HAF algorithm places the majority of cloudlets near the epicenter of the network where there are the most users. This strongly differs from the optimal placement seen in Fig. 7b, where cloudlets are more evenly distributed in the network. In cloudlet placement by the HAF algorithm, some users at the edge of the network have a distance of up to three AP hops before reaching their closest cloudlets, where as in the optimal placement, each user is no more than a single AP hop away from its closest cloudlet. The highly clustered placement of cloudlets produced by the HAF algorithm also underlines a weakness in the Closest-User-to-Cloudlet assignment, the outer layers of the cloudlet cluster will absorb most of the offloading requests from users in the outer regions of the network, while the inner cloudlets may receive much fewer user task requests. An example of this can be seen in Fig. 7c, the cloudlet at p_j only serves its own users, as user requests from the outer network are blocked by the surrounding cloudlets. This can lead to imbalanced workloads among cloudlets, resulting in an SRT even worse than that in the random placement. On the other hand, the DBC algorithm avoids clustering cloudlets together, by removing users at the AP of the cloudlet location at the end of each placement iteration. As a result, the cloudlet placement delivered by the DBC algorithm is much closer to the optimal one, with only a difference of one cloudlet placement site.

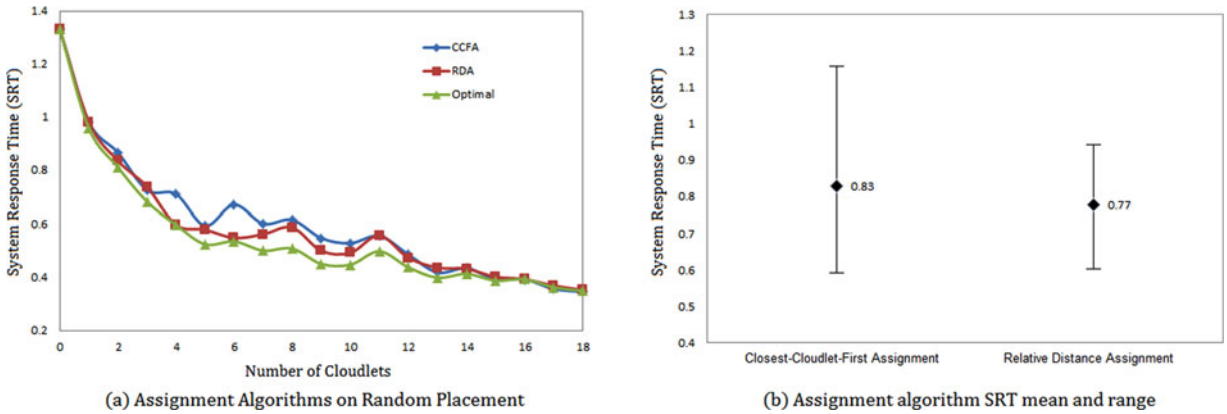


Fig. 8. System response time achieved by user-to-cloudlet assignment.

5.2.2 Performance Evaluation of User-to-Cloudlet Assignment Algorithms

The workload of user requests in a WMAN usually is highly volatile, while the cloudlet placement is static. As a result, it is important that the proposed user-to-cloudlet assignment algorithm consistently delivers low SRTs when the underlying cloudlet placement is incongruent with user demands. We now evaluate the two proposed user-to-cloudlet assignment algorithms, the closest-cloudlet-first assignment algorithm (CCF), and the relative distance assignment algorithm (RD).

In Fig. 8a, we look at the proposed assignment algorithms when given a random cloudlet placement in a network. On average, the closest-cloudlet-first assignment achieves an SRT 9 percent greater than the optimal assignment's SRT, whereas the relative distance assignment achieves an SRT of 7 percent on average. It can be seen that in the range $4 \leq K \leq 11$, the RD assignment is slightly better than the closest-cloudlet first assignment due to its load balancing heuristic.

The RD assignment usually assigns users to their closest cloudlets, and only occasionally assigns users to their alternative cloudlets when necessary. This allows the RD assignment to perform almost as well as the CCF assignment in most cases, and outperform the CCF assignment when cloudlet resources are limited, and cloudlet placement is less than ideal. With the growth of K, the CCF once again approaches the optimal performance as a surplus of cloudlets makes load balancing less important.

In Fig. 8b we look at the mean of the SRTs delivered by the two proposed user-to-cloudlet assignment algorithms, when three cloudlets have been placed, and the user population is randomly distributed within the network. We repeated the simulation 100 times, each with a new randomly generated user population to simulate snapshots of users as they move around the network. We recorded the maximum and minimum SRTs for both algorithms to show the value range of SRTs delivered by both algorithms. As can be seen, although the mean SRT delivered by the RD assignment algorithm is only marginally smaller than that of the CCF assignment algorithm, the range of the delivered SRTs is much smaller, indicating that the RD assignment algorithm is less sensitive to user mobility compared to the CCF assignment algorithm.

In summary, the mentioned case studies have shown a number of things. First, a strategic placement of cloudlets at APs is very important to reducing the system response time. Both the HAF and DBC algorithms outperform the random placement benchmark. By visually examining the placements, there are evidences to suggest that a well distributed cloudlet placement is preferable to a dense cluster of cloudlets at the network epicenter. Second, there appears to be a point at which the placement of additional cloudlets in the network follows the law of diminished returns. Finally, while the CCF assignment is effective when applied on top of an optimal cloudlet placement, volatile user demands can lead to unbalanced workloads among the cloudlets. Our case study simulation results have suggested that the RD assignment is a robust alternative to the CCF assignment.

5.3 Performance in Randomly Generated Networks

To further evaluate the performance of the cloudlet placement algorithms, we study the impact of system parameters on the performance of the proposed algorithms. Simulation results are the average across 100 randomly generated networks.

In Fig. 9a, we examine how the SRT delivered by our algorithms changes with the growth of the number of mobile users, while the number of APs remains to be fixed. With the increase on the number of users, the SRTs of all the three algorithms increase, due to heavier work loads on them. After an inflection point at 32 users, the SRT begins to plateau. The inflection point indicates that the cloudlets have become overloaded and are offloading a fraction of tasks from them to the remote cloud. Since we assume that the cloud has an infinite computing capacity, more tasks will be offloaded to the cloud when more and more users make use of the cloudlet services in the WMAN. Because the wait time of tasks at the cloud is constant, this causes the SRT of the WMAN to plateau the number of users in the WMAN increases.

In Fig. 9b, we compare the SRTs delivered by the two algorithms when normalized against the optimal SRT. The normalized SRT allows us to track the performance delivered by both algorithms in relation to the optimal one. The HAF and DBC algorithms begin with the performances of 44 and 65 percent respectively, and in general, the DBC algorithm significantly outperforms the HAF algorithm.

In Fig. 9c, we study the impact of the number of servers c on each cloudlet on the SRT of the proposed algorithms. With the growth of the number of servers in each cloudlet,

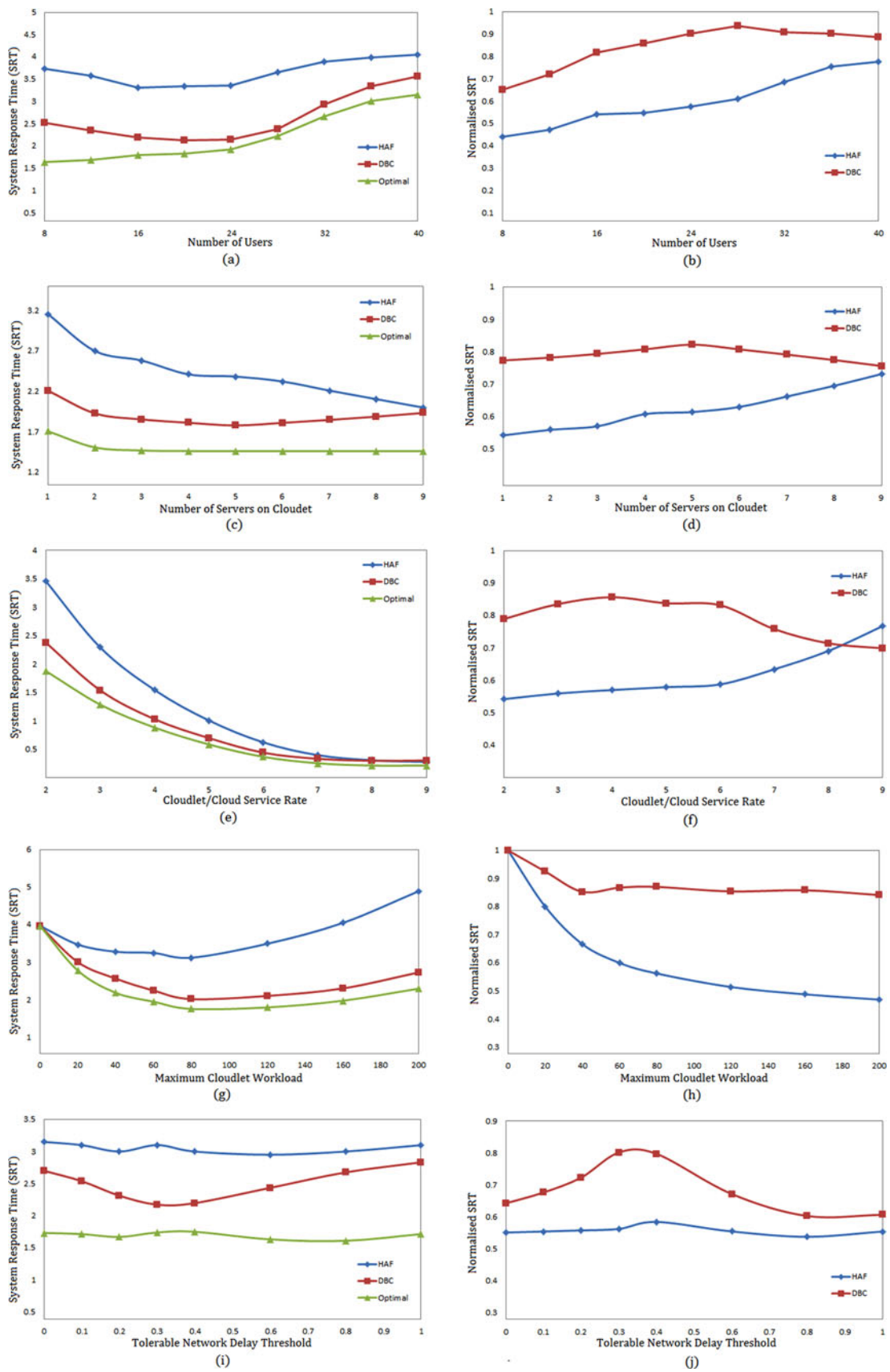


Fig. 9. Results for simulation sets (a)-(j).

the SRT of the optimal solution drops and quickly plateaus after three servers. This is because increasing the number of servers only reduces the average amount of time a task spends in the cloudlet queue. Once the cloudlet queue time becomes negligible, adding more servers to each cloudlet has very little effect on the SRT.

In Fig. 9d, we compare the normalized SRT delivered by the two algorithms as we increase the number of servers per cloudlet. The performance of the HAF algorithm starts with 54 percent and increases to 73 percent of the optimal one, and reaching the same normalized SRT as the DBC algorithm. This is because as the number of servers increases, balancing user workload amongst the cloudlets becomes less important as cloudlets are capable of handling higher loads. As the HAF algorithm guarantees that each user is assigned to its closest cloudlet, the HAF algorithm may have an advantage over the DBC algorithm when cloudlet resources are abundant.

In Fig. 9e, we investigate the impact of the service rate μ of cloudlets and the remote cloud on the performance of the proposed algorithms. Similar to increasing the number of servers on a cloudlet, the increase on the computing capacity of the cloud/cloudlet will decrease the SRT of the WMAN. The system response time eventually becomes plateaus, as it bounded from below by network delay.

As seen from Fig. 9f, the DBC algorithm begins with a normalized SRT of 79 percent, but slowly decreases to 70 percent. The HAF algorithm, on the other hand, begins with only a normalized SRT of 54 percent, but with the increase of cloudlet computing capacity, its performance eventually overtakes that of the DBC algorithm, which is similar to the results in Fig. 9c, as the cloudlet server capacity increases, each cloudlet has more than sufficient computing capacity to handle its workload, load balancing becomes less important. The DBC algorithm may be at a disadvantage compared to the HAF algorithm, as it occasionally assigns users to far away cloudlets to maintain the average workload balance across the cloudlets. The HAF algorithm, on the other hand, always sends user requests to their closest cloudlets, thereby minimizing the network delay.

Fig. 9g studies the impact of the maximum cloudlet load λ_{max} on the performance of the proposed algorithms. Initially, when $\lambda_{max} = 0$, the cloudlets offload all tasks to the cloud, resulting in all the three algorithms with the same SRT. As the value of λ_{max} increases, the SRTs by the DBC algorithm and the optimal one decrease to a local minimum at roughly $\lambda_{max} = 80$, and then steadily increase again when λ_{max}. From this, it can be seen that setting an optimal maximum cloudlet workload is an important issue. That is , a maximum workload that is too strict will limit the benefit of the cloudlet to users, while an unrestricted maximum workload may cause the SRT to be excessively long.

In Fig. 9h we compare the normalized SRT delivered by the two algorithms with the growth of λ_{max}, from which it can be seen that the DBC algorithm maintains a steady performance of roughly 82 percent, while the performance of the HAF algorithm monotonically decreases. This suggests it becomes even more important to assign mobile users to cloudlets evenly with the increase of λ_{max}.

Fig. 9i studies the impact of the tolerable network delay threshold T_{net} on the performance of the proposed algorithms. As T_{net} is only used in the DBC algorithm, increasing T_{net} has no effect on the optimal algorithm and the HAF algorithm. When $T_{net} = 0$, the set of candidate users for each AP is limited to the users wirelessly connected to the AP. As a result, the DBC algorithm has exactly the same cloudlet placement as the HAF algorithm when $T_{net} = 0$. Despite this, it can be seen that the DBC algorithm still delivers a better SRT than that of the HAF algorithm, indicating that the DBC algorithm has a more robust user-to-cloudlet assignment. The SRT delivered by the DBC algorithm decreases and reaches a local minimum when $T_{net} = 0.3$. It then steadily increases with the increase of the value of T_{net}, reflecting a decline in delivered performance. It is evident that with the increase of T_{net}, the set of candidate users for each AP will eventually include all the users in network. At this point the placement of cloudlets will essentially be random. As a result, finding an appropriate T_{net} is an important concern when applying the DBC algorithm to the KCP problem.

To conclude, we can expect the DBC algorithm to deliver a lower SRT than the HAF algorithm in most network environments. However, as seen in Fig. 9g, the HAF algorithm can outperform the DBC algorithm occasionally if the cloudlet computing capacity is disproportionately greater than the user workload arriving at the cloudlet. As such, the HAF algorithm may be recommendable for a small WLAN setting, where there are only a limited number of users and there is an emphasis on premium quality of service. However in most WMAN settings where cloudlets are a public utility, we can expect service providers to be more conscious of placing cloudlets in a cost-effective way. As a result, the proposed DBC algorithm is more appropriate for cloudlet placement as well as user-assignment to APs in WMANs, due to its consistently high performance.

6 Conclusion

Cloudlets are an important technology that provide performance improvement to many mobile applications. So far, very little attention has ever been paid to cloudlet placement in WMANs. In this paper we showed that the strategic placement of a limited number of cloudlets in a WMAN can significantly improve the performance of mobile user applications. We presented an efficient algorithm for solving the cloudlet placement problem. We also conducted experiments through simulation to evaluate the performance of the proposed algorithms. The simulation results demonstrated that the proposed algorithm is very promising. As this paper mainly focused on the system response time related to the positions of the cloudlets to be placed, there are other important issues that need further investigations in future. For example, the relationship between the total cost of cloudlet placement and the system response time. There are two main costs associated with cloudlets: the cost of their deployments and the cost of their maintenance/replacement. Since the technology applied to cloudlets in some degree is similar to the one applied to remote clouds, intuitively, it is reasonable to assume that cloudlets will have similar costs to that of cloud services. However, there are key differences between them (such as scale) that will need to be explored in order to derive an accurate estimate for cloudlet costs. Furthermore, the cloudlet performance has already been examined by many researchers in the past several years, most of the studies concentrated only on one cloudlet and small number of mobile users. In reality, the user scalability is an important issue. For example, what is the impact of the number of mobile users on the performance of cloudlets, and can cloudlets adjust their

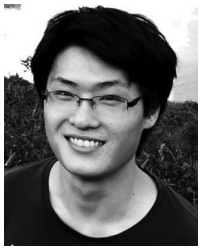
workloads quickly with dynamic demands of mobile users? On the other hand, there is still room for the improvement of the proposed system model. The current one cannot capture user movements within the network. Users in real-life WMANs are highly mobile, which can result in rapidly changes of the workloads amongst cloudlets. The improved system model is expected to be able to follow and predict user movement within the network.

Acknowledgments

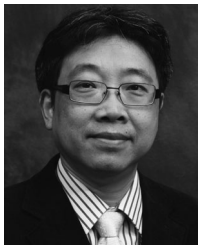
The authors appreciate the two anonymous referees for their constructive comments and valuable suggestions, which help improve the quality and presentation of the paper.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.
[2] 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," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2012, pp. 122–127.
[3] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient customization of mobile computing infrastructure," in *Proc. 1st Workshop Virtualization Mobile Comput.*, 2008, pp. 37–41.
[4] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl. Serv.*, 2013, pp. 153–166.
[5] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. Mobile Comput., Appl. Serv.*, 2012, pp. 59–79.
[6] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *Proc. IEEE 1st Int. Conf. Cloud Netw.*, 2012, pp. 80–86.
[7] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.
[8] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl. Serv.*, 2010, pp. 49–62.
[9] 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," in *Proc. INFOCOM*, 2012, pp. 945–953.
[10] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl. Serv*, 2011, pp. 43–56.
[11] M. Shiraz, S. Abolfazli, Z. Sanaei, and A. Gani, "A study on virtual machine deployment for application outsourcing in mobile cloud computing," *The J. Supercomput.*, vol. 63, no. 3, pp. 946–964, 2013.
[12] V. Cardellini, V. De Nito Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. Lo Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, 2013.
[13] W. Cai, V. C. Leung, and M. Chen, "Next generation mobile cloud gaming," in *Proc. IEEE 7th Int. Symp. Serv. Oriented Syst. Eng.*, 2013, pp. 551–560.
[14] W. Cai, V. C. Leung, and L. Hu, "A cloudlet-assisted multiplayer cloud gaming system," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 144–152, 2014.
[15] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 42–53, Jan.-Mar. 2014.
[16] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Serv.*, 2012, pp. 29–36.
[17] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Leveraging cloudlets for immersive collaborative applications," *IEEE Pervasive Comput.*, vol. 12, no. 4, pp. 30–38, Oct.-Dec. 2013.
[18] I. 802, "Ieee standard for local and metropolitan area networks: Overview and architecture," *IEEE Std 802–2014*, p. 8, 2014.
[19] B. Liu, *Theory and Practice of Uncertain Programming*. New York, NY, USA: Physica-Verlag, vol. 239.
[20] G. Sa, "Branch-and-bound and approximate solutions to the capacitated plant-location problem," *Oper. Res.*, vol. 17, no. 6, pp. 1005–1016, 1969.
[21] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. Hoboken, NJ, USA: pp. 101–103, 1975.
[22] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, Oakland, CA, USA, 1967, vol. 1, no. 14, pp. 281–297.
[23] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.
[24] Free wi-fi access in hong kong. [Online]. Available: http://www.discoverhongkong.com/au/plan-your-trip/practicalities/communications/wi-fi.jsp, Oct. 2014.



Mike Jia received the BSc degree in mathematics and computer science from Imperial College London in United Kingdom in 2013, and the honours in computer science at the Australian National University in 2014. He is a first year PhD student in computer science at the Australian National University. His research interests include mobile cloud computing and software defined networks. He was briefly a research assistant in the Department of Computing at Hong Kong Polytechnic University in 2013.



Dr. Jiannong Cao received the BSc degree in computer science from Nanjing University, Nanjing, China, and the MSc and Ph.D degrees in computer science from Washington State University, Pullman, WA, USA. He is currently a chair professor and a head of the Department of Computing at Hong Kong Polytechnic University, Hong Kong. He is also the director of the Internet and Mobile Computing Lab in the department. His research interests include parallel and distributed computing, wireless sensor networks, mobile and pervasive computing, and fault tolerance, covering protocols and algorithms, system platforms and middleware, and real-world applications. He has co-authored four books, co-edited nine books, and published more than 300 papers in major international journals and conference proceedings. He has directed and participated in numerous research and development projects and, as a principal investigator, obtained more than HK$25 million grants from government funding agencies and industries. He has received numerous prizes and awards, and is very active in professional activities locally and internationally. He is a senior member of China Computer Federation, a senior member of IEEE, and a member of ACM. He was the co-ordinator in Asia and now the chair of the Technical Committee on Distributed Computing of IEEE Computer Society. He has served as an associate editor and a member of the editorial boards of many international journals, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Networks*, *Pervasive and Mobile Computing Peer-to-Peer Networking and Applications*, and *Journal of Computer Science and Technology*. He has also served as a chair and member of organizing/program committees for many international conferences, including PERCOM, INFOCOM, ICDCS, IPDPS, ICPP, RTSS, DSN, ICNP, SRDS, MASS, PRDC, ICC, GLOBECOM, and WCNC. He is a fellow of the IEEE.

Weifa Liang (M'99–SM'01) received the BSc degree from Wuhan University, China in 1984, the ME degree from the University of Science and Technology of China in 1989, and the PhD degree from the Australian National University in 1998, all in computer science. He is currently an associate professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of routing protocols for wireless ad hoc and sensor networks, cloud computing, graph databases, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.