A Low-Cost Edge Server Placement Strategy in Wireless Metropolitan Area Networks

无线城域网低成本边缘服务器配置策略

Yongzheng Ren*, Feng Zeng*, Wenjia Li†, Lin Meng‡
*School of Software, Central South University, Changsha, China
†Department of Computer Science, New York Institute of Technology, New York, USA
‡Department of Electronic and Computer Engineering, Ritsumeikan University, Kyoto, Japan
email: {forstudy, fengzeng}@csu.edu.cn, wli20@nyit.edu, menglin@fc.ritsumei.ac.jp

***Abstract*—With the emergence of computation-intensive applications such as face recognition, natural language processing, etc., the performance of mobile devices appears to be weak. Mobile cloud computing is extensively used to cover the aforementioned shortage. However, the distance between remote cloud and mobile devices might be too far to be tolerable. Cloudlet computing, fog computing, and mobile edge computing are proposed to bring the computing resources near to the mobile devices to reduce the high delay caused by the long distance. Although there exist tremendous studies focused on the offloading strategies, where and how to place edge servers to minimize the edge server providers cost is seldom involved. In this paper, we first investigate the problem of minimizing the number of edge servers while ensuring QoS constraints such as access delay, and the Integer Linear Programming (ILP) formulation of this problem is given. Then, we transform it into the minimum dominating set problem of graph theory and propose a greedy algorithm to solve it. The simulation results demonstrate the proposed algorithm is promising.**

***Index Terms*—edge server placement, mobile edge computing, minimum dominating set**

I. Introduction

Nowadays, portable devices such as mobile phones, tablets, laptops and their applications have become an inseparable part of our lives which greatly improve the quality of our lives. However, with the increasing amount of data generated by these applications, mobile devices cannot efficiently process these data locally due to their limited battery life, storage and CPU capacity. A promising approach to bridge the gap is offloading some computational-intensive tasks from mobile devices to remote cloud whose computing resources are abundant. Nonetheless, the distance between mobile users and remote cloud be far, and the latency caused by it would be intolerable. On the other hand, plenty of requests may result in network congestion and eventually make the user experience terrible. In order to solve the aforementioned problems, cloudlet computing [1], fog computing [2] and mobile edge computing [3] have been proposed to be the complements of mobile cloud computing [4] whose main ideas are to bring computation and storage resources to the edge of the network. In this paper, we call those devices with rich resources (e.g., a cloudlet) edge servers. Mobile devices can offload all or part of the assignments to them, and they just need to return the execution result.

Most studies usually assumed that edge servers they need have been completely deployed and pay attention to the problems of computing offloading and resource allocation. However, without optimal edge server placement methods, these jobs can only be castles in the air. So, it is of great importance to investigate the edge server placement problem in order to establish a foundation for other work in mobile edge computing. Although there has been a few research on the placement of edge servers aiming at minimizing the average access delay between users and edge servers, they didn't take the budget of edge server providers into consideration. The more the number of servers, the better the user experience and the ideal situation is that there are edge servers co-located with each AP, which is impractical because the budgets of server vendors are always limited. The cost of deploying edge servers is mainly related to two factors [5]: site rentals and computation demands. The former factor means that the more locations are selected to deploy edge servers, the higher the cost is. The latter factor tells us that the greater the computation demand, the greater the number of edge servers which will result in more cost. In addition, the cost of edge server maintenance and management such as labor cost will increase as the number of edge servers increases. At the beginning of a project, the vendors often do not have enough funds to support the best service, but to save as much as possible to meet the basic needs of users. For example, an edge server provider is planning to deploy the MEC systems to serve all the population in a city. Unfortunately, the capital of the edge server provider is limited which means that it is impossible to deploy edge servers everywhere. The urgent matter is to economize as much as possible while ensuring the requirements of users, such as latency. Therefore, it is nontrivial to study how to save the cost of the edge server vendors.

边缘服务器的布局问题

In this paper, we consider the problem of minimizing the number of edge servers in a Wireless Metropolitan Area Network (WMAN) consisting of plenty of wireless Access Point (APs) while ensuring the QoS requirements of users such as access delay which could be regarded as an instance of more general facility location problem. There are two main challenges in this problem, one is how to choose the least number of APs co-located with an edge server to provide all users with good service, and the other is how to choose the

978-1-5386-5156-8/18/$31.00 ©2018 IEEE

edge servers to execute the tasks of users. Motivated by the gateway placement work of Aoun et al. [6], we present an efficient algorithm to divide the WMAN into clusters with radius constraint where the AP co-located with an edge server is the head of the cluster and APs in the coverage of the cluster should offload the tasks to the cluster head. The main contributions of this paper are as follows. We first study the problem of minimizing the number of edge servers in a WMAN and we formulate it as an ILP problem. Due to the poor scalability of the ILP, we transform it into the minimum dominating set problem of graph theory. Moreover, we extend the definition of dominating set and devise a greedy-based algorithm for the edge server placement problem. Finally, we evaluate the performance of the proposed algorithm by experimental simulations.

The remaining parts of this paper are organized as follows. Related works are reviewed in Section 2. The network model and ILP formulation are presented in Section 3. A minimum dominating set algorithm is proposed in Section 4. Simulation results are provided to evaluate the performance of the proposed algorithm in Section 5. Section 6 concludes the paper and future works.

II. Related Work

With the rise of mobile edge computing, cloudlet computing and fog computing, the research works on them emerge in an endless stream. One of the most important functions of these technologies is to offload the tasks to the edge servers to save energy and speed up the process of computation. Therefore, there exists a large quantity of research regarding computing offloading. This problem could be divided into several sub-problems, such as whether to offload or not, what should be offloaded and what the order of offloaded task should be, etc. If the local devices are able to conduct the tasks, just execute locally. Due to the size of data need to be transferred and the latency caused by the transmitting process, it should be considered that whether to offload full tasks consisting of composable components or to offload part of them. Moreover, when what should be offloaded is determined, the sequence of these tasks is also vital to ensure that it returns the correct results. Authors in [7] presented an aggregate network utility optimization framework for the problems of resource allocation. Zhang et al. [8] proposed an efficient code partition algorithm to help make decisions for offloading. Taking the dependency of offloaded components into consideration, Mahmoodi et al. [9] made decisions for offloading according to not only components need to be offloaded but also the scheduling order of these components. Zhang et at. [10] proposed a proactive service framework to minimize the response delay. Mach et al. [11] surveyed the current research work and classified them into three types: decision on computing offloading, allocation of computing and mobility management. Although there has been a number of research on these three hot spots, where edge servers should be placed namely the edge server placement problem has been neglected which is significant too.

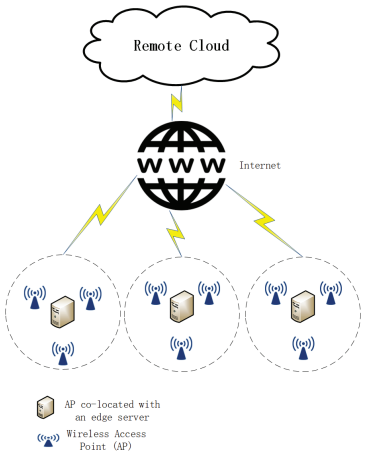

Fig. 1. Mobile Edge Computing Architecture.

To date and to the best of our knowledge, there are only a few works focused on the edge server placement problem. Satyanarayanan et al. [12] explored how application location impacted user experience, but they didn't discuss where edge servers should be placed in a WMAN context. Xu et al. [13] first formulated a novel capacitated cloudlet placement problem that place K cloudlets to strategic locations to minimize the average cloudlet access delay between the mobile users and the cloudlets in WMAN and showed that the problem is NP-hard. Then, they proposed serval efficient algorithms to solve it. Considering the redundant sorting process of Xu's algorithms, the authors in [14] proposed a new heuristic algorithm and a Particle Swarm Optimization (PSO) algorithm which outperform the existing algorithms. Moreover, Jia et al. [15] presented an efficient algorithm to deploy a limited number of cloudlets in a WMAN while balancing their workload. Meng et al. [16] raised an algorithm based on the snapshots of the network to place the edge servers in permanent locations and an online job routing algorithm to minimize the communication delay. Given the IoT networks, authors in [17] applied Software-Defined Networking (SDN) techniques to provide flexible and programmable management for cloudlet placement and proposed a ranking-based near-optimal placement algorithm. Taking the cost of deploying edge servers into consideration, Fan et al. [18] proposed a cost aware cloudlet placement strategy to optimize the tradeoff between the deployment cost and end-to-end delay. In view of the mobility of mobile users, movable cloudlets and a novel cloudlet placement method for mobile applications over GPS big data were introduced in [19].

Most of the above-mentioned works just pay attention to minimizing the average delay between edge servers and mobile users ignoring that the funds are limited. Considering the limited budgets of edge server vendors, this paper studies the problem of minimizing the number of edge servers while ensuring the QoS constraints such as latency.

III. System Model

In this section, we present our network model and formulate the problem. The architecture of mobile edge computing (MEC), consisting of remote cloud, Internet, APs and edge servers is shown in Fig. 1. We divide the WMAN into disjoint clusters (the circles made up of dashed lines in Fig. 1.) which comprise different numbers of APs. It is in the cluster heads that we decide to deploy edge servers. Note that the radiuses of the clusters are limited and different. For any AP with computation tasks need to be offloaded, the edge server within the cluster is the first choice, and the edge servers are connected by the Internet in order to prevent failures. Furthermore, the remote cloud is another choice for the offloaded tasks.

A. Network Model

We consider the problem of edge server placement in the context of WMAN. Given a connected undirected graph $G = (V, E)$ where V is the set of APs and E is the set of links between two APs. There is an edge $(u, v) \in E$, $u \in V$, $v \in V$ if and only if AP u and AP v are connected by a communication link. Denote $n = |V|$ as the number of APs and $m = |E|$ as the number of links between APs. Each AP could be selected as the location of an edge server. In other words, the edge servers are co-located with part of APs instead of selecting some other places which might cause additional delay. For simplicity, we assume that each edge server co-located with the AP has enough ability to meet the computing requirements within its own cluster and each hop between two APs has the same latency. Moreover, the distance between the node u and the node v, denoted by $d(u, v)$, is the minimum number of hops between them. Therefore, the delay between two APs is proportional to the distance between them. Denote $D(u, v)$ as the delay between AP u and AP v. $D(u, v)$ is proportional to $d(u, v)$. There are usually two ways to represent graphs, one is adjacency matrix, and the other is adjacency list. We use the widely used adjacency matrix to represent graph $G = (V, E)$ in which the rows and columns are labeled by the graph vertexes V with the number 1 or 0 in position (m, n) according to whether v_m and v_n are connected by any communication link.

B. Problem Formulation

在保证QoS约束的同时最小化边缘服务器数量

In this paper, we address the problem of minimizing the number of edge servers while ensuring the QoS constraints. The precise definition of the problem is described as follows. 问题定义：Given a WMAN $G = (V, E)$ and a delay upper bound D_{max}, our object is to find the minimum subset $K \subset V$ which could connect all APs in V with the D_{max} constraint. We define a binary variable x_i, $i \in V$, to represent whether an edge server is deployed at AP i (i.e., $x_i = 1$) or not (i.e., $x_i = 0$) and another binary variable $y_{(i,j)}$, $j \in K$, to denote whether the workload of AP i is assigned to edge server j (i.e., $y_{(i,j)} = 1$) or not (i.e., $y_{(i,j)} = 0$). Then, we use $c(j)$ and $r(i)$ to denote the capacity of edge server j and computation demand of AP i respectively. Recall that $d(u, v)$ is the minimum hops between AP u and AP v, and the delay between AP u and AP v, $D(u, v)$, is proportional to $d(u, v)$. Thus, the delay constraint D_{max} can be converted to the distance constraint d_{max}. It can be formulated as an Integer Linear Programming (ILP). Our object function is formulated as follows:

$$min \sum_{i \in V} x_i$$

Subject to

$$\sum_{i \in V} y_{(i,j)} \cdot r(i) \leq c(j), \forall j \in K \tag{1}$$

$$\sum_{j \in K} y_{(i,j)} \cdot d(i, j) \leq d_{max}, \forall i \in V \tag{2}$$

$$\sum_{j \in K} y_{(i,j)} = 1, \forall i \in V \tag{3}$$

$$x_i \in \{0, 1\}, \forall i \in V \tag{4}$$

$$y_{i,j} \in \{0, 1\}, \forall i \in V, j \in K \tag{5}$$

$$c(j) > 0, \forall j \in K \tag{6}$$

$$r(i) \geq 0, \forall i \in V \tag{7}$$

Constraint (1) ensures that each edge server has enough capacity to process all the computation tasks offloaded to it and inequality (2) ensures that the distance between each AP and the corresponding edge server is no more than the distance upper bound as to satisfy the requirement of delay. Constraint (3) denotes that each AP can only be assigned to one edge server. Note that inequality (1) represents the computation demand constraint and inequality (2) is the delay constraint of the QoS constraints.

IV. Minimum Dominating Set Algorithm

To minimize the number of edge servers while ensuring QoS constraints is similar to the minimum dominating set problem which is a classical NP-hard problem. However, there are some differences between them. Recall that D_{max} is the upper bound of access delay which users can tolerate and we assumed that the access delay in each hop is same for convenience. So, the access delay constraint can be transformed to the distance constraint which means we can use d_{max} to represent the constraint of access delay. In the case of $d_{max} = 1$, the problem of the edge server placement can be transformed into the problem of computing the minimum dominating set of a given graph. But it is not available when d_{max} is more than 1. So, we discuss the edge server placement problems in two situations and start with the condition of $d_{max} = 1$. The condition of $d_{max} > 1$ is discussed subsequently.

A. One Hop Constraint

When the distance the users can tolerate is one hop, namely $d_{max} = 1$, we use the traditional concept of dominating set to solve the problem. The definition of dominating set is given as follows.

Definition 1: A dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D.

From definition 1, we can know that any node in G is either in D or one hop away from some node in D. It is in D that we decide to place the edge servers. Let $G = (V, E)$ be the WMAN, our object is to compute the minimum dominating set D of G. For each AP v_i, there exists an edge server co-located with itself or one hop from it which ensures the access delay between APs and edge servers cannot violate the requirements of users. The access delay is smallest when users are in the region of an AP co-located with an edge server. Before describing the details of the multi-hop condition, we first introduce several concepts in the graph theory. The degree of a vertex in a graph is the number edges incident to the vertex and the degree of AP v_i is denoted by $deg(v_i)$. Then, let $nei(v_i)$ be the set of APs connected to v_i directly.

B. Multi-Hop Constraint

The traditional dominating set is not applicable when the delay requirement is more than one hop, namely $d_{max} > 1$. Therefore, we extend the definition of dominating set as follows.

Definition 2: A i-tier dominating set of a graph $G = (V, E)$ is a subset D^i of V such that for every vertex not in D^i there exist a link between the vertex and at least one member of D^i within i hops.

According to definition 2, the access delay constraint is extended to multi-hop condition. It is worth noting that when the value of i is 1, D^1 is equal to the traditional dominating set D. Since the concept of dominating set is extended, we extend the degree concept too. Recall that $d(u, v)$ is the minimum number of hops between u and v. And we define $extdeg(v, i)$ as the number of nodes that $d(v, u)$ is no more than i. When the value of i is 1, the value of $extdeg(v, i)$ is equal to the degree of v. Given a graph G and the tier of extended dominating set i, we select an AP which has the biggest value of $extdeg(v, i)$ as the location of an edge server. This means that each selection will cover the APs which has not been assigned to an edge server as many as possible so that the number of edge servers is minimum. Meanwhile, the distance from each AP to the edge servers is limited to i hops which guarantees that the access delay is tolerant. For the detailed computing process, we propose a Greedy-based Extended Dominating Set (GEDS) Algorithm as follows.

As is shown in Algorithm 1, we use the greedy approach to compute the minimum extended dominating set D^i of graph G. The for loop from line 6 to 10 selects a node with the biggest value of extdeg as the location for deploying the edge server from the set of uncovered nodes U in each while loop. Moreover, the cluster S is built after the selection process. The cluster S consist of cluster head s and the nodes covered by s, namely nodes whose value of $d(u, s)$ is no more than i. In addition, the shortest paths between s and the nodes covered by s are recorded in S too. As we mentioned before, the nodes in S should offload the computation tasks to the cluster head s which ensures that the distance between all nodes in S and s is no more than i hops. This means that all users can access an edge server in a short time.

Algorithm 1 Greedy-based Extended Dominating Set Algorithm

Input: A graph of the WMAN G and the tier of DS i;
Output: Locations of edge servers, D^i;

1: $U \leftarrow V$; /*V is set of nodes in G, U is the set of nodes uncovered by D*/
2: $C \leftarrow \emptyset$; /*C is the set of nodes covered by D */
3: $D \leftarrow \emptyset$; /*D is the locations of edge servers */
4: **while** U **do**
5: $s \leftarrow v_1$;
6: **for** each $v \in U$ **do**
7: compute the distance between v and all other vertexes in U, denoted by d(v,u)
8: compute the number of the nodes that d(v, u) is no more than i, denoted by extdeg(v)
9: **if** $extdeg(v, i) > extdeg(s, i)$ **then**
10: $s \leftarrow v$;
11: **end if**
12: **end for**
13: $S \leftarrow s \cup nei(s)$;
14: save S as a cluster, s is the cluster head
15: $D \leftarrow D \cup s$;
16: remove the nodes in S from G;
17: $U \leftarrow U - S$;
18: $C \leftarrow C \cup S$;
19: return D^i;
20: **end while**

C. An example

In this section, we show an example of above-mentioned algorithm. Fig. 2 shows a graph consisting of 70 nodes in an area of 10×10. Our object is to find the minimum dominating set and the minimum 2-tier dominating set. Then, the GEDS algorithm is applied to compute the minimum dominating set D, namely D^1, and the minimum 2-tier dominating set D^2. In Fig. 3, solid circles represent the nodes of the D, solid rectangles denote the nodes of D^2 and the nodes belong to both D and D^2 are the solid diamonds. We can see that the number of nodes in D is 15, the number of nodes in D^2 is 5 and the number of common nodes is 3. If the graph can represent a WMAN in reality, we just need 15 edge servers to provide basic services for the whole WMAN when users can bear the delay caused by one hop. In addition, if the quality of the communication link is wonderful and users can bear the delay caused by two hops, we only need 5 edge servers to cover all the WMAN. In other words, only seven percent

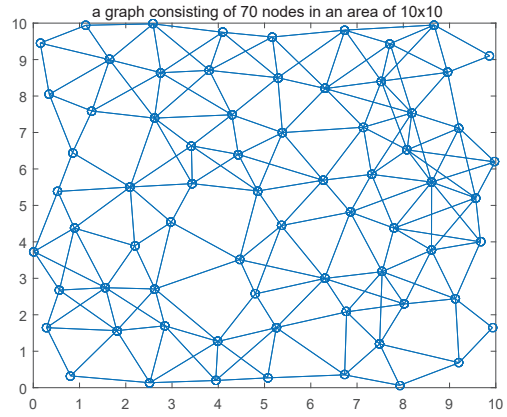

Fig. 2. a graph consisting of 70 nodes in an area of 10x10

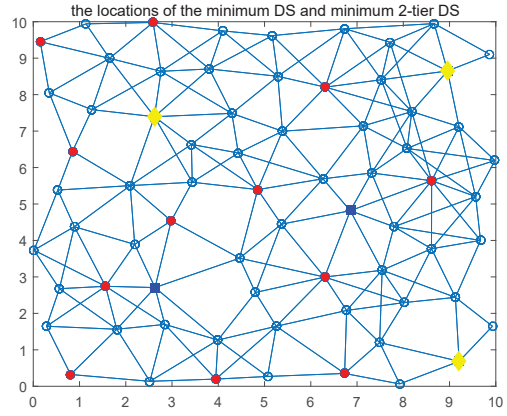

Fig. 3. the locations of the minimum DS and minimum 2-tier DS

of APs need to deploy an edge server which can save a lot of maintenance and management costs for edge server providers.

V. Experimental Analysis

In this section, we present simulation results to evaluate the performance of the proposed GEDS algorithm. We first introduce the method of random topology generation. Then, we evaluated the performance of GEDS according to the number of required edge servers.

A. Random Topology Generation

It is great that if we can apply our algorithm to a real WMAN topology. Unfortunately, the data of WMAN topology is often managed by the government and not open to the public. Thus, we need to generate random topologies to model the real WMANs for our experiments. In this paper, we just control the density and the number of APs of a WMAN due to the complexity of modeling a real WMAN. Let γ denote the coverage of all the APs and we assume that there exists a link between two APs if the distance between them is less than γ. In addition, let δ represent the minimum distance between two APs which is consistent with the fact that two APs cannot be too close. Given a random area of $Mkm \times Mkm$, the number of APs N, γ and δ, we generate the random topology step by step. First, we create an AP at a random location in the area. Then, we add a new random AP to the area. Two conditions we use to control the density of APs must be met if the new added AP stays in the area, one is that the AP must in the coverage of the existing AP, the other is the distance between them cannot exceed γ. If the new generated AP cannot meet these two conditions, it will be deleted and we will regenerate one. The process of adding a new AP to the area will stop when the number of APs reaches N.

B. Performance Evaluation 性能评估

We evaluate the performance of GEDS algorithm by varying the network size and delay constraint respectively. For reliability, each experiment was conducted 20 times and we use the average of them to report our performance results.

We first evaluate the performance of GEDS algorithm against the Random algorithm by varying the network sizes. As the name suggests, the Random algorithm places edge servers to APs randomly until the whole network is covered. Given the area of $30km \times 30km$, delay constraint is 2, $\gamma = 2$ and $\delta = 1$, we change the number of APs from 100 to 500. As we can see from Fig. 4, the number of edge servers increases with the increase in the number of APs which is consistent to the fact that the more APs, the more edge servers we need and our algorithm outperforms Random algorithm. Particularly, the number of servers has decreased by nearly half when the network size is 100. In addition, the figure shows that although the number of edge servers is increasing as the network size increases, its growth is slowing down. This means that as the size of the network expands, the number of additional edge servers that need to be added is less and our algorithm is effective.

Then, we show the experiment result of different delay constraints while fixing the network size. Same as the last section, we describe the network details firstly. The area of the network is still $30km \times 30km$ and the number of APs is 100. The values of γ and δ are equal to the previous part. Fig. 5 shows that with the growing of delay constraint the number of APs decreases. It means that the longer the users can bear, the fewer edge servers we need. Although GEDS algorithm has always been better than Random algorithm, the gap between them is smaller as the number of hops increases. This is because as the number of hops increases, the locations of edge servers become less important. However, most of applications have the requirements of ultra-low delay. In addition, the access delay is related to the quality of the communication link and the size of the offloaded tasks. So, the number of hops users can tolerate will not be large in a real scenario.

VI. Conclusion and Future Works

There is only a small mount of research on the edge server placement in WMAN and most of them ignore the budget limitation of edge server vendors. In this paper, we focused on saving the cost of deploying edge servers to serve all the APs in a WMAN and we described it as the problem of minimizing the number of edge servers while ensuring QoS constraints.

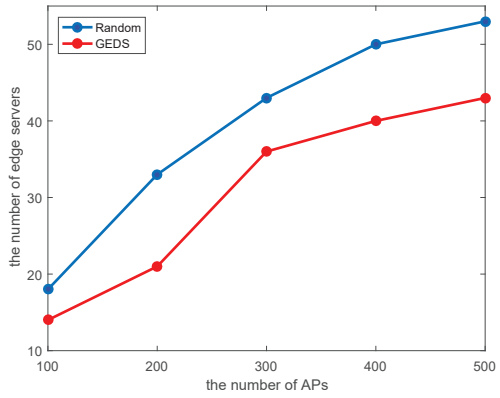

Fig. 4. The number of edge servers with different network sizes

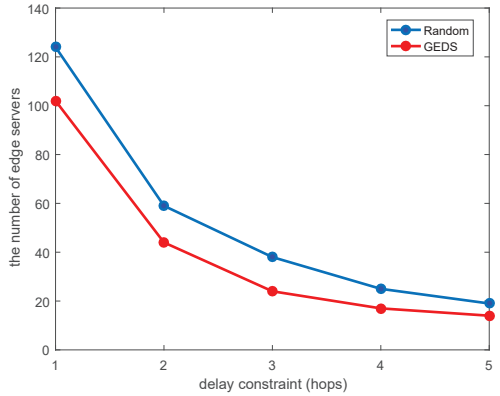

Fig. 5. The number of edge servers with different delay constraints

Then, we presented an ILP formulation for the edge server placement problem. Due to the poor scalability of the ILP, we transformed the problem into the minimum dominating set problem of graph theory. Furthermore, a greedy-based algorithm was proposed to compute the minimum dominating set. We extended the definition of dominating set and proposed a corresponding algorithm to compute the minimum extended dominating set. Finally, we evaluated the performance of the proposed algorithm by experimental simulations. The simulation results show that the proposed algorithm is promising. For our future work, it is interesting to study the load balancing problem to make sure all edge servers work well.

REFERENCES

[1] Satyanarayanan M, Bahl P, Cceres R, et al. The Case for VM-Based Cloudlets in Mobile Computing[J]. IEEE Pervasive Computing, 2009, 8(4):14-23.

[2] Bonomi F, Milito R, Zhu J, et al. Fog computing and its role in the internet of things[C]. Edition of the Mcc Workshop on Mobile Cloud Computing. ACM, 2012:13-16.

[3] Shi W, Cao J, Zhang Q, et al. Edge Computing: Vision and Challenges[J]. IEEE Internet of Things Journal, 2016, 3(5):637-646.

[4] Mell P, Grance T. The NIST definition of cloud computing[J]. Communications of the Acm, 2011, 53(6):50-50

[5] Mao Y, You C, Zhang J, et al. A Survey on Mobile Edge Computing: The Communication Perspective[J]. IEEE Communications Surveys & Tutorials, 2017, PP(99):1-1.

[6] Aoun B, Boutaba R, Iraqi Y, et al. Gateway Placement Optimization in Wireless Mesh Networks With QoS Constraints[J]. IEEE Journal on Selected Areas in Communications, 2006, 24(11):2127-2136.

[7] Zhang D, Chen Z, Awad M K, et al. Utility-Optimal Resource Management and Allocation Algorithm for Energy Harvesting Cognitive Radio Sensor Networks[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12):3552-3565.

[8] Zhang Y, Liu H, Jiao L, et al. To offload or not to offload: An efficient code partition algorithm for mobile cloud computing[C] IEEE, International Conference on Cloud NETWORKING. IEEE, 2013:80-86.

[9] Mahmoodi S E, Uma R N, Subbalakshmi K P. Optimal Joint Scheduling and Cloud Offloading for Mobile Applications[J]. IEEE Transactions on Cloud Computing, 2016, PP(99):1-1.

[10] Zhang D, Shen R, Ren J, et al. Delay-optimal Proactive Service Framework for Block-Stream as a Service[J]. IEEE Wireless Communication Letters, 2018, PP(99):1-1.

[11] Mach P, Becvar Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading[J]. IEEE Communications Surveys & Tutorials, 2017, PP(99):1-1.

[12] Clinch S, Harkes J, Friday A, et al. How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users[C]. IEEE International Conference on Pervasive Computing and Communications. IEEE, 2012:122-127.

[13] Z. Xu, W. Liang, W. Xu, M. Jia and S. Guo, "Efficient Algorithms for Capacitated Cloudlet Placements," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 10, pp. 2866-2880, Oct. 1 2016.

[14] L. Ma, J. Wu, L. Chen and Z. Liu, "Fast algorithms for capacitated cloudlet placements," 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD), Wellington, 2017, pp. 439-444.

[15] M. Jia, J. Cao and W. Liang, "Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks," in IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 725-737, Oct.-Dec. 1 2017.

[16] J. Meng, W. Shi, H. Tan and X. Li, "Cloudlet Placement and Minimum-Delay Routing in Cloudlet Computing," 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Chengdu, 2017, pp. 297-304.

[17] Zhao L, Sun W, Shi Y, et al. Optimal Placement of Cloudlets for Access Delay Minimization in SDN-based Internet of Things Networks[J]. IEEE Internet of Things Journal, PP(99):1-1.

[18] Q. Fan and N. Ansari, "Cost Aware cloudlet Placement for big data processing at the edge," 2017 IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-6. doi: 10.1109/ICC.2017.7996722

[19] Xiang H, Xu X, Zheng H, et al. An Adaptive Cloudlet Placement Method for Mobile Applications over GPS Big Data[C]. Global Communications Conference. IEEE, 2017:1-6.