Mobility-aware dynamic service placement for edge computing

Gang Liu, Jiawei Wang, Yumin Tian*, Zhenhua Yang, Zhenping Wu

School of Computer Science and Technology, Xidian University, 710071 Xi'an, China.

Abstract

Edge computing provides capabilities similar to traditional cloud computing at network edge closer to users. Offloading applications from the cloud to the edge of the network can effectively diminish the latency of users waiting for applications to improve their quality of service(QoS). Due to the user mobility, however, the distance between the user and the edge device is constantly increasing, resulting in a degradation of the user's QoS. To satisfy the QoS of mobile users, applications should be dynamically migrated to follow the user mobility.In this paper, we propose a mobility-aware dynamic service placement policy that reduces the number of virtual machine migrations and diminishes user-perceived latency simultaneously. The proposed scheme filters out the invalid migration that the user handoffs to another access point before the migration is completed. The simulation trials demonstrate that our solution improves the QoS of users and reduces the number of application migrations.

Received on 30 May 2019; accepted on 01 July 2019; published on 29 July 2019

Keywords: Edge computing, quality of service, dynamic service placement, user mobility

doi:10.4108/eai.13-7-2018.163922

1. Introduction

With the rapid growth of the number of delay-sensitive applications in the Internet of Things (IoT), traditional cloud data centers where IoT applications are placed cannot meet the delay requirements of these applications since data need to be transferred over long-distance from the cloud to the end-users (EU). A long transmission distance increases the response time and bandwidth consumption, which leads to increasing service latency and decreasing application availability. To solve these issues, some investigators proposed to add an abstraction stratum, usually referred to as the fog stratum or the edge stratum, between the cloud and the EUs or end-devices. This stratum provides computation, storage, and communication resources to offload applications from the cloud to the network edge closer to EUs, which can reduce the latency of users waiting for services. In this case, the response time of the application is shortened and the bandwidth consumption of the long distance is reduced since the application is fulfilled at network edge closer to users, which means that the user's QoS and application availability are improved.[1]

While cloud-based service placement schemes are widely studied, it is not feasible to simply apply cloud-based placement schemes to edge computing environments. This is because the goal of service placement and application environment are different for the most part. The goal of service placement in cloud computing can be roughly divided into three categories: (1) *services consolidation*,[2] which aims to consolidate the services to a smaller set of physical servers to maximize resource utilization; (2) *performance improvement*,[3] which tries to place the services to a set of heterogeneous nodes to take advantage of the heterogeneities on energy efficiency or cost efficiency; (3) *load balancing*,[4] which seeks to place the services to different nodes according to the working status of the nodes in terms of workload. However, the main goal of service placement in edge computing is to improve the QoS, such as reducing the user-perceived latency. Furthermore, the service placement in edge computing requires taking into account the limited resources of the edge device, which is different from cloud devices with abundant resource. Therefore, it is very meaningful

*Yumin Tian. Email: ymtian@mail.xidian.edu.cn to design the optimized service placement scheme for the special scenario of edge computing. Several studies cope with service placement and resource optimization in the scenarios of edge computing.[5–7] These efforts focus on placing services closer to users under limited resources of fog devices in order to diminish the user-perceived latency.

However, the papers above did not address the problem of migration once users' location changed. Due to the user mobility, EUs will cause wireless handover when they move out of the coverage area of an access point (AP). The distance between the user and the edge device on which the application is placed is constantly increasing, which degrades the user's QoS. Service migration caused by the user mobility has been addressed in some papers. Filiposka et al[8] proposed a service migration pattern that services should be migrated among fog nodes to follow the user mobility. However, it is worth noting that frequent migration may incur additional migration costs, such as time latency and energy consumption. Therefore, it is necessary to reduce the number of migrations as much as possible while satisfying users' QoS, such as reducing the user-perceived latency.

The objective of this paper is to design a dynamic service placement policy that takes into account the migration cost of frequent migration and improves QoS for EUs. Specifically, we first explore how to place services under the limited resources of fog nodes to minimize user-perceived latency. Second, we construct a model of the sojourn time of the EUs at AP and the use of this model reduces invalid migration so as to improve the user's QoS. The work in this paper is based on previous studies,[8] but please note that we propose a different virtual machine (VM) migration scheme that takes into account the cost of migration.

Figure 1 illustrates the scenario proposed in this paper. When a mobile user requests a service within the coverage of the AP A_1 at a certain time, it is obvious that if we want to minimize the latency perceived by users, the service should be placed on the edge node connected by A_1. Considering the mobility of the user, the mobile user needs to pass through the areas covered by the A_2 and A_3. We assume that the user's sojourn time in A_2 is less than the application migration time from fog node connected by A_1 to fog node connected byA_2. Namely, the user has moved out of the coverage of A_2 before the migration is completed, so this migration is not helpful for reducing user-perceived latency and we consider it as an invalid migration. We can see from Section 3 that the user-perceived latency ultimately depends on the migration latency of the service in our problem. The presence of these additional invalid migrations degrades the user's QoS. This paper improves the user's QoS by predicting the sojourn time of a user on the AP, reducing the number of invalid migrations.

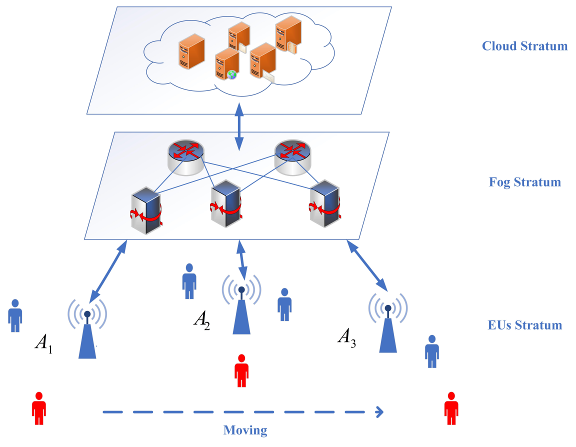


Figure 1. An example of dynamic service placement.

Dynamic service placement in edge computing has been addressed in the previous studies.[8, 9] However, we have considered some issues that were rarely noticed in previous studies. The main contributions of this paper are summarized as follows:

- The migration of services requires additional costs. Frequent service migrations will increase the user-perceived latency and cause network congestion. How to reduce the number of migrations on the premise of satisfying users' QoS is studied in this paper.
- A dynamic service placement algorithm based on the sojourn time in each coverage of AP is proposed. Inspired by [10] and [11], the user mobility is built as a model of the sojourn time of the end-users in each coverage of APs, following an exponential distribution. The proposed algorithm reduces the user-perceived latency by reducing invalid migrations that the migration time is longer than the user's sojourn time.

The rest of this paper is organized as follows. Section 2 reviews related work. System models and problem formulation are presented in Section 3. The dynamic service placement algorithm which consists of initial placement of VM and VM migration algorithm is described in Section 4. The results from our simulation are discussed in Section 5. Finally, Section 6 concludes this paper.

2. Related works

For the special environment of edge computing, the problem of the optimized service placement has been studied using different methods.Complex networks,[6] Greedy algorithm,[7] Integer linear programming,[12] Matching Game,[13] Mixed Integer

Linear Programming[14] and Genetic algorithm[15] are the common solutions in this field. For instance, a service placement policy in Fog computing based on graph partitions is proposed,[6] aiming to improve the service availability and QoS. The author builds the fog stratum as a dendrogram using graph partition method and places the application as far as possible to the device community with higher depth that has enough available resources to meet the demand of an application and that includes the gateway where the current user is connected. Placing applications inside device communities with higher depth can effectively improve the service availability even in the event of line failures. The results from their experimental scenarios show their strategy significantly shortened the application placement time and effectively improved the service availability and QoS. But the author assumes that the users are unalterably connected to the same gateway or AP. The mobile users and services migration are ignored. In the work of Souza et al,[7] the problem of service allocation is modeled as a multidimensional knapsack problem in order to diminish the allocation delay. The balance between adaptive load and energy consumption is dynamically established by the conditions of the system. Similarly, the solution did not address the problem of migration once users' location changed.

In order to satisfy QoS, services should be migrated to follow the user mobility. Several studies[9, 16, 17] migrate applications in advance by predicting the user mobility to reduce the user-perceived latency. For instance, in the work of Nademnega et al,[16] the authors propose a mobility-based service migration prediction scheme. The aim in the proposed prediction scheme is to obtain the tradeoff between the overhead and latency. To achieve this goal, this solution predicts data transfer throughput, handoff time and VM migration management in advance. However, it should be noted that the experimental results of these schemes are based on the fact that they accurately predict the user's future information, which is generally irregular and unpredictable. As accurately predicting user mobility is not feasible in practice, Tao et al[18] researched the mobile edge service performance optimization problem and considered the cost of frequent migration. The author uses Lyapunov optimization to decompose long-term optimization problems into a series of real-time optimization problems without requiring some prior knowledge such as user mobility.

In the work of Filiposka et al,[19] the authors first introduced the concept of community to optimize resource management in fog computing and proposed that VMs be allocated to a set of fog devices using the community relationship of the devices and migrated as users move. They improved the previous scheme in follow-up work.[8] The authors implemented a the "Follow me" VM migration, but did not take into account the operational costs of frequent migrations. This scheme is somewhat similar to the one proposed in this paper, but we consider the problem of migration cost and reduce the invalid migration by predicting user sojourn time. The comparison of the results will be presented in Section 5.

3. System model

In this paper, we consider the hierarchical infrastructure of the edge as shown in Figure 1. It usually consists of three stratums: cloud stratum; fog stratum; EUs stratum. Some models in this paper are based on previous studies. For example, VM placement constraints and migration delay model in the work of Lera et al[6] are used to describe our system.

Let $\text{E} = (1, 2, ..., E)$ denote the set of edge devices. An edge device $i \in \text{E}$ is defined by the available capacities of its resources S_i, which is a vector that typically contains physical attributes of a machine, such as the number of CPU, the size of RAM. Container migration is much more difficult than VM migration.[20] VM migration is used in this paper. A VM j is defined as the number of resources it requires RS_j. Similar to the definition of edge devices, it is usually a vector. Here we take a binary indicator to denote the service placement variable. Let $a_{ij} = 1$ if VM j is placed in on the edge device $i \in \text{E}$, and $a_{ij} = 0$ else. Note that the placement of VMs is limited by the resource capacity of the edge device. Therefore, the resource required for the assigned VMs should be less than the maximum resource available of the edge device.

$$\sum_{j=1}^{N} a_{ij} \times RS_j \le R_i \quad \forall i \in \text{E}. \tag{1}$$

In edge computing, the user-perceived latency is determined by computing delay and migration delay. Edge devices are also represented by processing capacity MIPS, measured by millions of instructions per second. For the sake of simplicity, we assume that all edge devices in our scenario are isomorphic. It is also easy for researchers to deploy heterogeneous edge devices to our scenario. When an application is identified, the latency perceived by the user depends primarily on the migration delay. The migration delay usually consists of propagation delay and transmission delay of the network. Thus, the migration delay $D_{i,j}$ between edge devices i and j can be calculated as follows:

$$D_{i,j} = PR_{i,j} + \frac{size}{BW_{i,j}}. \tag{2}$$

where $PR_{i,j}$ is the propagation delay, $BW_{i,j}$ represents the network bandwidth and *size* is the size of the VM to be transmitted.

Although the dynamic placement of applications can improve the QoS, please note that the migration of applications will bring additional costs. It is very meaningful to minimize the number of migrations as much as possible while ensuring the QoS for users. The user mobility is modeled as a sojourn time model within the coverage of an AP, which follows an exponential distribution.[10, 11] We use t_{uk} to represent the sojourn time of user u within the coverage of AP k.

$$f(t) = \frac{1}{\tau_{uk}} e^{-\frac{t}{\tau_{uk}}} \quad t \geq 0. \tag{3}$$

where τ_{uk} indicates the average sojourn time. A small τ_{uk} indicates that users frequently handoff between APs. Thanks to the rapid development of machine learning, we can get accurate τ_{uk} by collecting some information of users, which will be part of our future work. Here we assume that τ_{uk} follows the Gaussian distribution.[11]

We propose our optimization objectives based on the previous models above. From the analysis above, it can be seen that the user-perceived latency is determined by computing delay and migration delay. However, the computing delay is usually determined by the physical conditions of the device, such as RAM, MIPS, so optimizing the computing delay is not within our consideration. We mainly focused on the optimization of migration delays. This paper introduces the sojourn time model of the user at the AP to reduce invalid migration and thus reduce the user-perceived latency. Considering that there are M users, the total user-perceived latency can be expressed as follows:

$$T = \sum_{m=1}^{M} (T_m + \sum_{i=1}^{|\Omega_m|} D_{e_i, e_{i+1}}). \tag{4}$$

where T_m is the computing delay of the service requested by user m, Ω_m represents the ordered set of the edge devices where the VM requested by user m is placed sequentially before the completion of the user request service and e_i is the edge device in Ω_m.

In a word, regarding our dynamic service placement scheme, the goal is to minimize the latency perceived by users. In particular, the problem can be described as follows:

$$\begin{aligned} &\min \ T. \\ &s.t. \ (1). \end{aligned} \tag{5}$$

where our objective function is to minimize the delay perceived by users. Constraint (1) indicates that the resource required for the allocated VMs should be less than the maximum resource available of the edge device.

4. The mobility-aware dynamic service placement scheme

The dynamic service placement algorithm in this paper consists of two phases. First, the smallest community that has enough available resources to meet the user request and that connects the current AP in use is selected to instantiate a VM. Second, we mainly dealt with the problem of migration of edge resources due to the user mobility. The first phase of initialization of VM placement problem has been studied,[8] based on which we put forward a VM migration scheme.

4.1. Initial placement of edge resources

Considering the limited computation, storage, and communication resources of the edge device, the available resources provided by a single edge device cannot meet the resources required to instantiate a VM. The introduction of the concept of community can solve the problem of limited resources of a single edge computing device. By using community detection algorithms,[21] the edge infrastructure is divided into a series of overlapping community structures that are usually composed of more closely connected sets of edge devices. These device communities provide an abundant pool of virtual resources compared to a single edge device. The problem of resource placement in edge environments is a NP-hard problem that involves selecting the optimal network entity to deploy user applications.[22] In our problem, the optimal device community is chosen to deploy the VM requested by the user.

Algorithm 1 Algorithm 1 Community-based VM Placement

```
Initialization: Input network infrastructure topology.
    Calculate device communities $\mathbb{C}$ using community
    detection algorithms.
End initialization.
 1: $currentAP \leftarrow$ obtain the AP the user is currently using.
 2: $VM \leftarrow$ obtain the VM that needed to be placed.
 3: $Comms \leftarrow$ find the device community that connects
    the $currentAP$ from $\mathbb{C}$ and sort them from bottom to top.
 4: for $comm$ in $Comms$ do
 5:     if hasEnoughResources($VM$,$comm$) then
 6:         placeVMInDevice($VM, comm$)
 7:         updateResource($VM, comm$)
 8:         Break ;
 9: if allocationFailInEdge($VM$) then
10:     placeVMInCloud($VM$)
```

The VM initialization placement algorithm[8] is used in the first phase of our algorithm. According to the results of the community detection algorithm, such as a hierarchical dendrogram, assigning VMs to edge devices with sufficient resources to deploy and execute a VM instance can be performed in the following two steps: 1) Search for the lowest-level community that connects the AP that the user is currently using and that has sufficient resources to deploy the VM requested by the user;2) Assign VMs to edge devices of the chosen community using some traditional placement algorithm. Algorithm 1 shows the pseudocode of the initial placement of edge resources. More implementation details of the algorithm can refer to the work of Filiposka et al.[8]

4.2. Migration of edge resources

Due to the user mobility, the user may move out of the service coverage of the original AP and handoff to a new AP that is served by a different set of edge devices. In order to ensure that the user's QoS is not degraded, the migration process should be triggered when the user handoffs to a new AP attached to a different device community. Considering the existence of invalid migration, that is, the time that the user stays at the new AP is less than the migration delay of the VM, then this migration is not beneficial to improve user's QoS. Because before the migration is completed, the users are likely to leave the coverage of the new AP and handoff to a different AP. As mentioned earlier, such an invalid migration increases the migration delay and degrades the QoS, such as the user-perceived latency.

The pseudocode of the VM migration scheme proposed in this paper is presented in Algorithm 2. It is worth noting that even if all the destination communities connected to the *newAP* do not have enough resources, we will not place the VM in the cloud compared to the initial placement scheme but forward the request to the origin community where the original VM was placed. This is because the cost of data transmission in the fog stratum is much less than the cost of communication between the fog stratum and the cloud.[23]

In Algorithm 2, we first determine whether the AP that the user is using and the AP that the user last connected is connected to the same edge gateway. If they are connected to the same edge gateway, the VM will not be migrated because even if the user handoffs APs, these APs are served by the same set of edge devices. The line 10 of the code predicts how long users will stay in the new AP, following an exponential distribution as mentioned in Equation (3). The relationship between migration delay and sojourn time is considered in line 11. When the user's sojourn time is less than the transmission delay, the user may leave the coverage of the new AP and handoff to a different AP before the migration is completed. Such migration increases the delay of transmission, so that the migration delay increases and degrades the user's QoS. In the scheme proposed in this paper, the VM migration program is triggered only if the sojourn time of a mobile user is more than the migration delay of VM.

Algorithm 2 Algorithm 2 Virtual machine migration scheme

```
1:  $oldAP \leftarrow$ obtain the AP the user is currently using.
2:  $newAP \leftarrow$ obtain the AP the user is currently using.
3:  $Comms \leftarrow$ find the device community that connects the $currentAP$from $\mathbb{C}$ and sort them from bottom to top
4:  if getEdgeGateway($newAP$) == getEdgeGateway($oldAP$) then
5:      Return ;
6:  for $comm$ in $Comms$ do
7:      if hasEnoughResources($VM$,$comm$) then
8:          $D_{i,j} \leftarrow$ Calculate the transmission delay of the VM.
9:          $t_{uk} \leftarrow$ Predict the sojourn time of user $u$ in $newAP$ $k$.
10:         if $D_{i,j} \leq t_{uk}$ then
11:             placeVMInDevice($VM$, $comm$)
12:             updateResource($VM$, $comm$)
13:             Break ;
```

5. Performance evaluation

5.1. Simulation setup

To validate and evaluate the mobility-aware dynamic service placement proposed in this paper, MyiFogSim[24] which is an extension of iFogSim[25] is used to conduct system simulation. MyiFogSim complements the shortcomings of iFogSim in supporting the user mobility. Some necessary functions have been added to simulate mobile user VM migration, such as AP, the VM migration technology, the migration strategy and so on.

To perform our simulation, a map with the total coverage area size of 1200×1200m is used. APs with a coverage range of 100m are evenly distributed on this map, resulting in a total of 144 APs. These APs are connected to the edge gateway closest to them. The results from our simulation scenarios show that the number of APs connected to each edge gateway is between 2 and 9. The upload delay between the AP and the edge gateway is set to 4ms. A total of 25 edge gateways evenly distributed in the map are generated and each edge gateway is connected to 10 physical hosts. To simplify the problem, we use a scalar to represent the resources owned by the host such as the size of the RAM. The RAM size of each physical host is set to 250MB. The bandwidth of the network link is set between 125MB and 250MB and the propagation delay of all network links is set to 0.5ms.

The virtual machine migration technology used in the migration process is based on on-live provided by MyiFogSim. The VM is firstly suspended for non-live migration.[26] Secondly, the running status will be wrapped and transmitted to the target site. All network connections are closed during the migration and are re-established after the VM is restored. The virtual machine has a processing capacity of 2000 MIPS. For the sake of simplicity, we denote the amount of resources required by the virtual machine as a scalar such as RAM. Of course, it is also very easy to extend it into a vector in our problem. We assume that the amount of RAM required by a virtual machine is 64. The size of the virtual machine is randomly selected between 800MB and 1600MB.

We use network usage and application performance in terms of latency as the criteria for evaluating schemes. By analyzing the source code of the MyiFogSim simulator and combining with the previous work[27], the calculations of the two criteria are expressed as follows:

$$Network\ usage = \frac{\sum\limits_{Req}\left(T_{Req}^{lat} \times Size_{Req}\right)}{T_{sim}}. \tag{6}$$

where T_{Req}^{lat} is the network delay between the original device of request and the target device of the request, $Size_{Req}$is the size of the request being sent and T_{sim} is the simulation time. In a word, the network usage is the sum of the network usages generated by each request Req within the simulation time T_{sim}.

In MyiFogSim, each application is modeled as a processing loop between its modules and the results can be obtained only if all services are completed. Service latency is measured as the average time to execute a set of interdependent services. So the latency is expressed as the time difference between the point in time when the first service of the application is started t_{start} and the point in time when the last service of the application is executed t_{end}.

$$Latency = \frac{\sum\limits_{Req}\left(t_{end} - t_{start}\right)}{|Req|}. \tag{7}$$

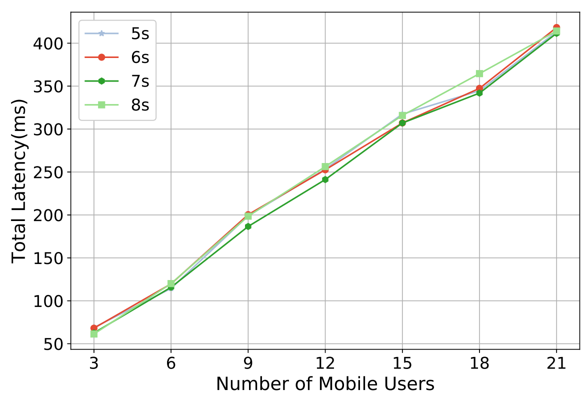
5.2. The impact of sojourn time on proposed scheme

In the scheme proposed in this paper, the sojourn time of the user is an important parameter that affects the experimental results. We explored the effects of different user's sojourn time on experimental results. As mentioned above, it is assumed that the average sojourn time of users follows a Gaussian distribution $N(\mu_t, \sigma_t^2)$.We explored the impact of user sojourn time on the proposed solution with $\sigma_t = 1s$ and $\mu_t = 5s, 6s, 7s, 8s$, respectively. In the experiment, in order to simulate the sojourn time of the user at different APs, the user's speed is dynamically changed at different APs. Once the user handoffs to a different AP, the sojourn time of the user at that AP will be predicted. According to the predicted time, a mobile user speed is dynamically calculated. Here it should be noted that although the speed of a user changes dynamically, the user's direction of movement remains the same.

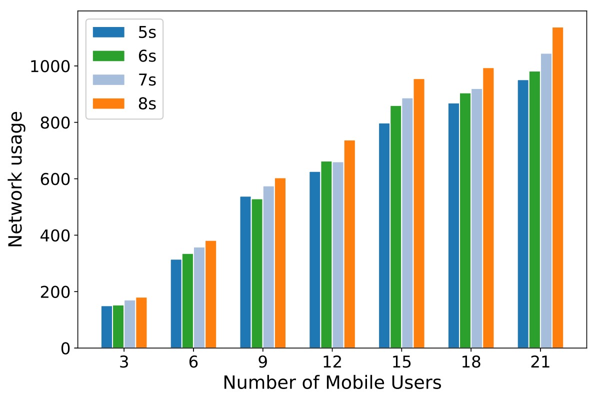
Due to the user mobility, the distance between the user and the edge device on which the application is placed is constantly increasing, which increases latency in data propagation so that the user's QoS is degraded. In order to satisfy the user's quality of service, VMs should be migrated to follow the user mobility. If the sojourn time of the user in an AP is not considered, the user may move out of the coverage area of that AP before the migration is completed. These invalid migrations increase the user-perceived latency.

Figure 2(a) shows that when the average sojourn time of the user is 7s, the total latency perceived by the user is the smallest. On the contrary, the total latency perceived by all users is maximized when the average sojourn time of users is 8s. This means that the user obtains the best quality of service when the average sojourn time is equal to 7s. When the average user residence time is greater than (or less than) 7s, the number of migrated VMs is greater than (or less than) the number of VMs migrated with an average migration time of 7s. Too many VM migrations that include many invalid migrations increase the VM migration delay and the users' QoS are degraded. The number of migrated VMs is too small to allow the VMs to follow the user mobility so that the latency perceived by the user also increases.

In Figure 2(b), the network usage increases with the increase of the average sojourn time of users when the number of users is the same. The longer the user stays, the more VMs are being migrated, resulting in a continuous increase in network usage. In particular, considering the extreme case where all the sojourn times are greater than the maximum transmission delay of the VM, the proposed scheme at this time is equivalent to the "Follow me" migration scheme. When the sojourn time of the user is small enough, for example, all sojourn times are less than the minimum transmission delay of the VM, all virtual machines will not be migrated because the sojourn time of the user is too small. The solution proposed in this case is equivalent to the "No Migration" scheme.



(a)



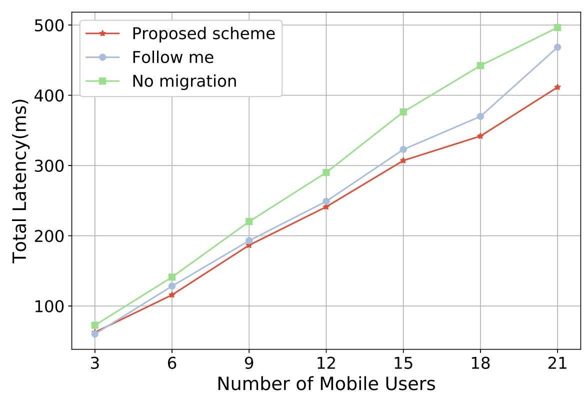
(b)

Figure 2. The impact of UEs' average sojourn time with different number of users.(a) Total service latency perceived by users. (b) Network usage.

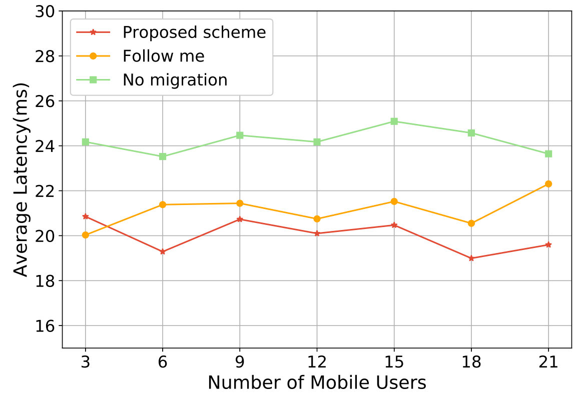
5.3. The comparison of different service placement schemes

In order to verify the performance of the proposed scheme, some existing schemes are implemented in the same simulation scenario. We considered two representative scenarios for service placement. The first scenario is where the VM always keeps the initial assignment position unchanged no matter where the user moves. The second scenario is that as long as the user handoff to a different AP that is served by a different set of edge devices, the service VM is always migrated to execute on service at the place closest to the user.[8] In the following, we will use "No migration" for the first scenario and "Follow me" for the second scenario.

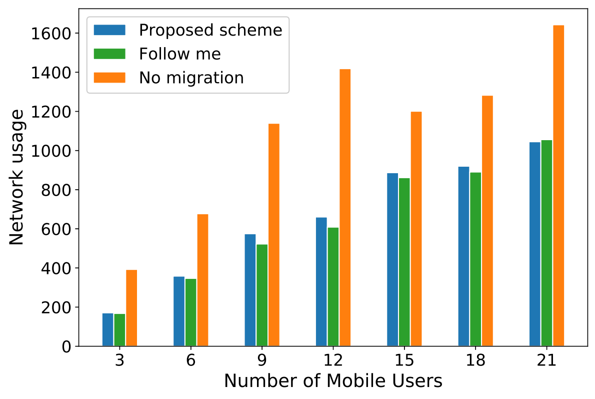
A comparison of the total latency perceived by users for different numbers of users is given in Figure 3(a). Without migration, the latency perceived by users is



(a)



(b)



(c)

Figure 3. Comparison of three different service placement policies with different number of users. The average user's sojourn time is 7s. (a) Total service latency perceived by users. (b)Average service latency perceived by users. (c) Network usage.

the greatest. The "Follow me" and the scheme proposed in this paper effectively diminish the user-perceived latency by migrating the service VM to the place closest to the user. Although the "Follow me" scheme migrates the VM to the edge device closest to the user to execute the request compared to the "No migration" scheme, there is still room for improvement. There may be some migrations in the "Follow me" scenario where the user handoff to another AP before the migration is completed. Such invalid migrations are not beneficial for improving the QoS. The proposed scheme makes these invalid migrations no longer migrate as the users move. Experimental results show that the proposed scheme minimizes the latency perceived by users. Figure 3(b) shows the average latency for different schemes. From the figure we can see that the average latency of the three different schemes are distributed around 24ms, 21ms and 20ms. The proposed scheme reduces the average latency of each application by 4ms compared with the "No migration" scheme; Compared with the "Follow me" scheme, the proposed scheme reduces the average latency by 1ms by reducing invalid migration. The proposed scheme reduces the number of VM migrations and diminishes the latency perceived by users simultaneously.

Figure 3(c) shows a comparison of the network usage of the three different schemes. The results show that the network usage of the "No migration" scheme is always far more than the other two service placement schemes. On the one hand, this is because network usage depends on network latency between edge devices according to Equation (6). On the other hand, the service VM always keeps the initial assignment position unchanged. Due to the user mobility, users will be farther and farther from the edge devices where the VM is initially placed. Long-distance data transmission increases the network load and the network usage. The migration of virtual machines follows the mobility of users. Although it will increase network usage of the transfer VM, the added overhead of virtual machine transmission is less than the overhead of long-distance data transmission according to the experimental results. The proposed scheme reduces the invalid migration, but it can be seen from Figure 3(c) that compared with the "Follow me" scheme, the network usage proposed by this paper has increased slightly. Please note, however, that the main goal of the proposed scheme is to reduce the latency perceived by users. Therefore, it is acceptable to sacrifice some network usage in our problem.

We further analyze the number of VM migrations and the number of user handoffs at different APs for the three different schemes. We define the migration ratio. The migration ratio is defined as the number of VM migrations divided by the number of user handoffs in a simulation experiment.

$$Migration\ ratio = \frac{the\ number\ of\ VM\ migrations}{the\ number\ of\ user\ handoffs}. \tag{8}$$

In Figure 4, the migration ratios for the three different scenarios are presented. In the strategy of "No migration", the VM migration ratio is 0. This is because the VM always keeps the initial assignment position unchanged. In the Follow me policy, the VM migrates as the user moves. The migration ratio at this time should ideally be 1 but the experimental results show that the migration ratio is distributed around 0.2. This is because the lowest-level device community connects multiple different APs. When the mobile user handoff these APs connected to the same edge gateway, the VM migration process will not be triggered. With the increasing number of users, the migration rate of the "Follow me" scheme will converge at around 0.2 and the proposed scheme will converge at 0.15. We have verified the performance of the proposed scheme with a small number of mobile users, but we can foresee that our scheme is still efficient even with a large number of users.

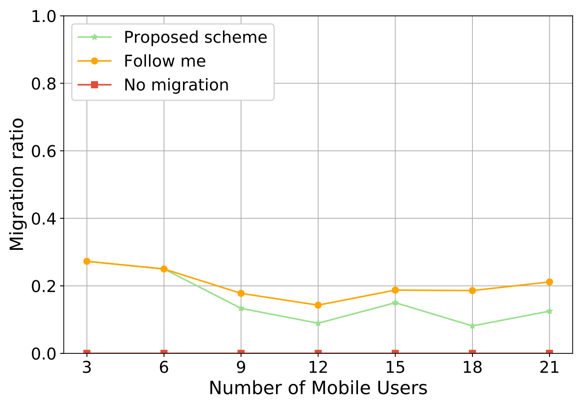


Figure 4. Comparison of the migration ratio with different number of users. The average user's sojourn time is 7s.

6. Conclusions

In this paper, we study the placement and migration of VM in edge computing environments under the constraints of limited edge device resources. We also research the operational cost of frequent migrations. Based on this, a mobility-aware dynamic service migration policy is designed to improve users' QoS and reduce the number of VM migrations simultaneously. Considering the migration delay of the VM and the sojourn time of the user within the coverage of an AP, the proposed scheme reduces the perceived service latency of users by filtering out invalid migrations. Simulation results show that the dynamic service placement algorithm proposed in this paper effectively improves the user's QoS compared with other schemes. However, the proposed scheme only considers the placement and migration of a VM and supposes that τ_{uk} follows the Gaussian distribution, which makes it unsuitable to be widely used in practice. More precisely, the fact is that the service in the scenario is often composed of multiple VMs and the sojourn time of the user is very complicated. Therefore, in our future work, the placement and migration of complex applications with multiple VMs will be studied. Moreover, to better apply the proposed scheme to practice, we will study how to obtain more reliable and accurate user sojourn time with the help of machine learning.

Acknowledgement

This work was partly supported by the Shaanxi Key R&D Program (Grant No. 2019ZDLGY13-01) and the Science and Technology Projects of Xi'an, China (Grant number: 201809170CX11JC12).

References

[1] Zhang J, Wang X, Huang H, and Chen S. Clustering based virtual machines placement in distributed cloud computing. *Future Generation Computer Systems* 2017; 66: 1–10.

[2] Jiang JW, Lan T, Ha S, Chen M, and Chiang M. Joint VM placement and routing for data center traffic engineering. In: IEEE. ; 2012: 2876–2880.

[3] Xu F, Liu F, Jin H, and Vasilakos AV. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE* 2013; 102(1): 11–31.

[4] Bharathi PD, Prakash P, and Kiran MVK. Energy efficient strategy for task allocation and VM placement in cloud environment. In: IEEE. ; 2017: 1–6.

[5] Jia M, Cao J, and Liang W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing* 2015; 5(4): 725–737.

[6] Lera I, Guerrero C, and Juiz C. Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Internet of Things Journal* 2018; 6(2): 3641–3651.

[7] Souza VB, Masip-Bruin X, Marin-Tordera E, Ramírez W, and Sanchez S. Towards distributed service allocation in fog-to-cloud (F2C) scenarios. In: IEEE. ; 2016: 1–6.

[8] Filiposka S, Mishev A, and Gilly K. Mobile-aware dynamic resource management for edge computing. *Transactions on Emerging Telecommunications Technologies* 2019; 30(6): e3626.

[9] Yang L, Cao J, Liang G, and Han X. Cost aware service placement and load dispatching in mobile cloud systems. *IEEE Transactions on Computers* 2015; 65(5): 1440–1452.

[10] Wang D, Liu Z, Wang X, and Lan Y. Mobility-Aware Task Offloading and Migration Schemes in Fog Computing Networks. *IEEE Access* 2019; 7: 43356–43368.

[11] Liu X, Zhang J, Zhang X, and Wang W. Mobility-aware coded probabilistic caching scheme for MEC-enabled small cell networks. *IEEE Access* 2017; 5: 17824–17833.

[12] Naas MI, Parvedy PR, Boukhobza J, and Lemarchand L. iFogStor: An IoT Data Placement Strategy for Fog Infrastructure. *In IEEE ICFEC.* ; 2017.

[13] Gu Y, Saad W, Bennis M, Debbah M, and Han Z. Matching theory for future wireless networks: fundamentals and applications. *IEEE Communications Magazine*; 53(5): 52-59.

[14] Ascigil O, Phan TK, Tasiopoulos AG, Sourlas V, Psaras I, and Pavlou G. On Uncoordinated Service Placement in Edge-Clouds. *In IEEE CloudCom* 2017: 41-48.

[15] Wen Z, Yang R, Garraghan P, Lin T, Xu J, and Rovatsos M. Fog Orchestration for Internet of Things Services. *IEEE Internet Computing* 2017; 21(2): 16-24.

[16] Nadembega A, Hafid AS, and Brisebois R. Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE. In: IEEE. ; 2016: 1–6.

[17] Aissioui A, Ksentini A, Gueroui AM, and Taleb T. On enabling 5G automotive systems using follow me edge-cloud concept. *IEEE Transactions on Vehicular Technology* 2018; 67(6): 5302–5316.

[18] Ouyang T, Zhou Z, and Chen X. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE Journal on Selected Areas in Communications* 2018; 36(10): 2333–2345.

[19] Filiposka S, Mishev A, and Gilly K. Community-based allocation and migration strategies for fog computing. In: IEEE. ; 2018: 1–6.

[20] Pahl C, and Lee B. Containers and clusters for edge cloud architectures–a technology review. In: IEEE. ; 2015: 379–386.

[21] Rosvall M, Axelsson D, and Bergstrom CT. The map equation. *The European Physical Journal Special Topics* 2009; 178(1): 13–23.

[22] Lera I, Guerrero C, and Juiz C. YAFS: A simulator for IoT scenarios in fog computing. *arXiv preprint arXiv:1902.01091* 2019.

[23] Mouradian C, Naboulsi D, Yangui S, Glitho RH, Morrow MJ, and Polakos PA. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials* 2017; 20(1): 416–464.

[24] Lopes MM, Higashino WA, Capretz MA, and Bittencourt LF. Myifogsim: A simulator for virtual machine migration in fog computing. *In: ACM.*; 2017: 47–52.

[25] Gupta H, Vahid Dastjerdi A, Ghosh SK, and Buyya R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience* 2017; 47(9): 1275–1296.

[26] Zhang F, Liu G, Fu X, and Yahyapour R. A survey on virtual machine migration: Challenges, techniques, and open issues. *IEEE Communications Surveys & Tutorials* 2018; 20(2): 1206–1243.

[27] Guerrero C, Lera I, and Juiz C. A lightweight decentralized service placement policy for performance

optimization in fog computing. *Journal of Ambient Intelligence and Humanized Computing* 2019; 10(6): 2435–2452.