IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 1, 2018, accepted April 15, 2018. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2018.2828102

A Survey on Service Migration in Mobile Edge Computing

SHANGGUANG WANG[1], (Senior Member, IEEE), JINLIANG XU[1], NING ZHANG[2], (Senior Member, IEEE), AND YUJIONG LIU[1]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Department of Computing Sciences, Texas A&M University at Corpus Christi, Corpus Christi, TX 78412, USA

Corresponding author: Shangguang Wang (sgwang@bupt.edu.cn)

This work was supported by NSFC under Grant 61472047.

ABSTRACT Mobile edge computing (MEC) provides a promising approach to significantly reduce network operational cost and improve quality of service (QoS) of mobile users by pushing computation resources to the network edges, and enables a scalable Internet of Things (IoT) architecture for time-sensitive applications (e-healthcare, real-time monitoring, and so on.). However, the mobility of mobile users and the limited coverage of edge servers can result in significant network performance degradation, dramatic drop in QoS, and even interruption of ongoing edge services; therefore, it is difficult to ensure service continuity. Service migration has great potential to address the issues, which decides when or where these services are migrated following user mobility and the changes of demand. In this paper, two conceptions similar to service migration, i.e., live migration for data centers and handover in cellular networks, are first discussed. Next, the cutting-edge research efforts on service migration in MEC are reviewed, and a devisal of taxonomy based on various research directions for efficient service migration is presented. Subsequently, a summary of three technologies for hosting services on edge servers, i.e., virtual machine, container, and agent, is provided. At last, open research challenges in service migration are identified and discussed.

INDEX TERMS Mobile edge computing, service migration, live migration, migration path selection, cellular handover.

I. INTRODUCTION

Cloud computing technology has been widely used in the past decade, which relies heavily on the centralization of computing and data resources, so that these resources can be accessed in an on-demand way by the distributed end users. Cloud services are provided by large centralized data-centers that may be located far away from the users. As a result, a user can endure long latency due to connection to remote services. In recent years, considerable progresses have been made to distribute cloud services closer to users, providing higher reliability and faster access at the same time.

Specifically, in Internet of Things (IoT) applications, to improve the data throughput and rapid response of mobile devices or sensors, a small cloud can be connected directly via the wireless communication infrastructure at the network edges (e.g., cellular base station and Wi-Fi access point) to provide services to the mobile users within its coverage. Mobile edge computing (MEC) can enable computation and data offloading for mobile devices [1]–[5], which is a supplementary for mobile devices with relatively limited computational and storage capacity. It is also useful in scenarios that require high data processing capability or robustness, e.g., in hostile environments [6] or in vehicular networks [7]. Many conceptual models have been proposed by academia and industry, including MEC [8], [9], mobile micro-cloud [10], micro datacenter [11], Cloudlet [12], Fog Computing [13]–[15], and Follow Me Cloud (FMC) [4]. These conceptual models are partially overlapping and complementary. The core of these models is to run applications and related processing tasks in proximity of mobile users, network congestion is reduced, battery life is enhanced and service experience is improved [16]. We use the term *Mobile Edge Computing* to refer to a general conceptual model and differentiate it from the above-mentioned models. In addition to significantly reducing network operational cost and improving quality of service (QoS) of mobile users by pushing computation resources closer to the network edges, MEC also enables a scalable IoT architecture for time

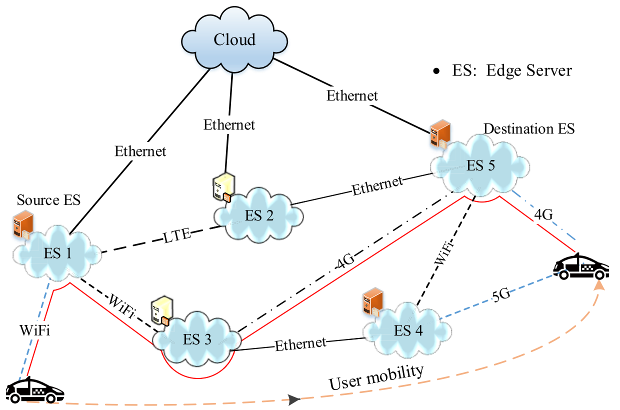


FIGURE 1. **A case of service migration in mobile edge computing. The red solid line means one transferring path between source and destination edge server.**

sensitive applications (e-healthcare, real time monitoring, etc.) [17]–[21].

MEC has emerged as a key enabling technology for realizing the IoT visions [19]. A significant issue in MEC is service migration with user mobility. The contradiction between the limited coverage of single *edge server* and the mobility of user terminals (e.g., smartphones [8] and intelligent vehicles [22]–[24]) will result in significant network performance degradation, which can further lead to dramatic drop in QoS and even interruption of ongoing edge services, therefore, it is difficult to ensure the service continuity [12], [25], [26]. Therefore, in order to ensure service continuity as users move, it is especially important to realize seamless service migration (i.e., without disruption of ongoing edge services, a mobile user is not allowed to freely move over a large geographic area). Since edge servers are attached to many different access points or base stations, a decision should be made that whether and where to migrate the ongoing edge services as an arbitrary user moves outside the service area of the associated edge server [27]. Considering the scenario as shown in Fig. 1, an edge server (e.g., a small cloud) contains one or more physical machines hosting several virtual machines, covers the mobile users in proximity. These edge servers are interconnected with each other via different kinds of network connections. Note that we use *edge server* as a general term to refer to the small cloud, such as cloudlet [12], fog node [13], [28], etc. In addition, we consider service migration as the stateful migration of applications: a mobile user accepts a service for a continuous time period, and the service application reserves internal state data for the user, such as intermediate data processing results. After the completion of the migration, the service resumes exactly where it stopped before migration. As a mobile user moves from one area to another, we can 1) either continue to run the service on the current edge server, and exchange data with a mobile user through the core network or other edge servers, 2) or migrate the service to another edge server that covers the new area. In both of the two cases, cost can be incurred: such as data transmission cost for the former case, and migration cost for the latter.

Service migration is also very challenging [4], [12], [25], [29]. When a user moves through several adjacent or overlapped geographical areas, service migration should deal with: 1) whether the ongoing service should be migrated out of the current edge server that hosts this service; 2) if the answer is yes, then which edge server the service should be migrated to; 3) how the service migration process should be carried out, considering the overhead and QoS requirements. This problem comes from the tradeoff of migration cost (e.g, migration cost and transmission cost)in the whole service migration process and improvement of users' expectation on QoS that can be achieved after migration (i.e., reducing the latency for users or network overhead). It is very difficult to obtain the optimal service mitigation because of the high uncertainty of user mobility and request patterns, as well as potential non-linearity of transmission and migration cost. Since edge servers are allocated at the network edges, their performance is intimately related to the dynamics of users. Moreover, service migration becomes more complex, considering a large number of users and applications, as well as the heterogeneity of edge servers.

In recent years, several survey papers have been published to provide overviews of the MEC area. These works mainly focus on system and network models, computation offloading, resource allocation, architectures and applications [5], [9], [19]. To the best of our knowledge, this is the first work that summarizes the problem of service migration in MEC. The contributions of this paper are: 1) review of the up to date research on service migration in MEC; 2) comparison with two similar concepts of service migration, i.e, live migration for data centers and handover in cellular networks; 3) devisal of taxonomy based on various research directions for efficient service migration; 4) summary of three hosting technologies of services on edge servers, i.e. virtual machine, container and agent; 5) identification of various open issues related to service migration which need further research.

The remainder of this paper is organized as follows. Section II presents two conceptions similar to service migration and a comparison between them. In Section III, we detail the techniques of migrating running services. In section IV, we discuss some of existing strategies of service migration. In Section V, we explore the pros and cons of three technologies for hosting mobile application components, i.e., virtual machine, container and agent. In Section VI, we discuss some research challenges in service migration. The main content is as shown in Fig. 2, each entry in frame corresponds to one section.

II. EXISTING CONCEPTS: SIMILARITY AND COMPARISON

In this section, we introduce two similar concepts that are closely related to service migration and compare them for better understanding of service migration.

A. LIVE MIGRATION FOR DATA CENTERS

Live migration of virtual machine is gaining more importance to improve the utilization of resources, load balancing of

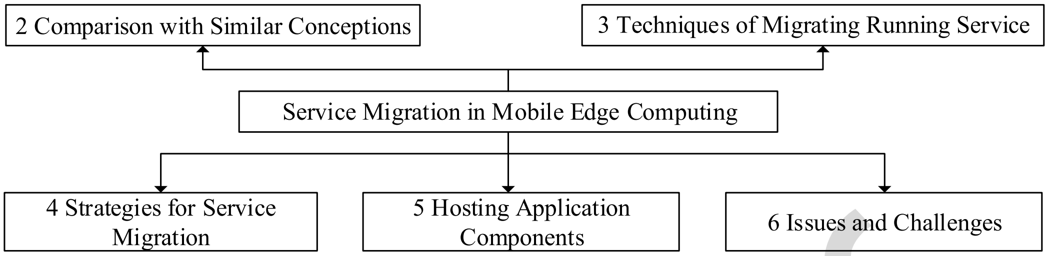


FIGURE 2. The organization of this survey. The number shows the corresponding section number. Sections 3, 4, 5 together constitute the body of the service migration topic. Among them, Section 5 is the lowest level of the topic, which describes the host of application components that need migration. If some running service is to be migrated, we should know what the corresponding application components are and what are hosting them. As a result, we say what Section 5 deals with is the lowest level, or the fundamental part. Section 4 describes the strategies in migrating the application components described in Section 5, which is a higher level topic. Section 4 and 5 are not enough for service migration as we must apply them into mobile edge computing network environment, that is what Section 3 is doing, e.g., how to reduce the data volume to be transferred. So there exists a progressive relationship between the three sections. But at the same time they deal with three different parts of and form the body of service migration.

processing nodes, tolerating the faults in virtual machines, etc., to increase the portability of nodes and to promote the efficiency of the physical server [30]–[34]. Live migration for data center mainly deals with memory migration of virtual machine instances. To transfer the memory state data of a virtual machine from its source physical machine to the destination machine, two techniques can be adopted, namely pre-copy and post-copy memory data migration.

1) In the former technique, all memory pages from the source to the destination are duplicated while the virtual machine instance is still running. If some pages change in the duplicating period, they will be copied again, until the ratio of re-copied pages is higher than the ratio of changed pages. After this phase, the instance on the source stops, the remaining changed pages are moved to the destination and the virtual machine instance resumes at the destination.
2) While post-copy memory migration is started by suspending the virtual machine instance on the source host. Then a minimal set of state data (including CPU state, register, non-pageable memory, etc.) is moved to the destination, then the instance is restarted on the destination.

Post-copy method transfers less data, but may incur long downtime. In contrast, pre-copy can reduce downtime, however, it needs transfer more data. Service migration in MEC resembles live migration in data centers, as they both try to move a runtime application from one virtual machine to another. However, they are at least in three important ways as follows [12]:

1) They target on different performance metrics. Service migration aims to reduce the total time of completion of migration, as end-to-end latency deteriorates until the end of the process. While live migration deals with the short period of the final step (i.e., downtime, during which mobile users cannot receive service), of which the total time is not the first consideration.
2) Live migration for data centers can make use of shared storage and memory, which are assumed to be very large and rich. While in MEC environment, these local resources are limited, this needs invoke application partition and task scheduling techniques.
3) The edge server deployment should accept whatever computation or network resources exist across geographically distributed edge servers. Different from live migration in data centers, service migration cannot depend on the availability of a dedicated computation unit or high-bandwidth network. As a result, service migration needs overcome high variation of network bandwidth and computation capacity caused by time-varying workload.
4) The required operating system and applications of the ongoing service may exist on the destination edge server. This can avoid unnecessary data transferring in service migration.

The distinction between live migration and service migration is as shown in Table 1.

TABLE 1. Distinction between live migration and service migration.

Field	Service migration	Live migration
Evaluation index	Total service migration time	Downtime time
Shared resource	No	Yes
Resource guarantee	No	Yes
Mirror image reuse	Yes	No

B. HANDOVER IN CELLULAR NETWORKS

In a cellular system, as the mobile user is moving across different cells during an continuous communication, handover (or handoff) needs to be performed [35]–[38], to avoid service interruption.

Similar to handover in cellular networks, service migration also deals with user mobility from one geographical area to another. However, they are different in the following aspects:

1) The data transferred in handover process of cellular networks contains signal messages and state data between a pair of mobile terminal and base station, or two base

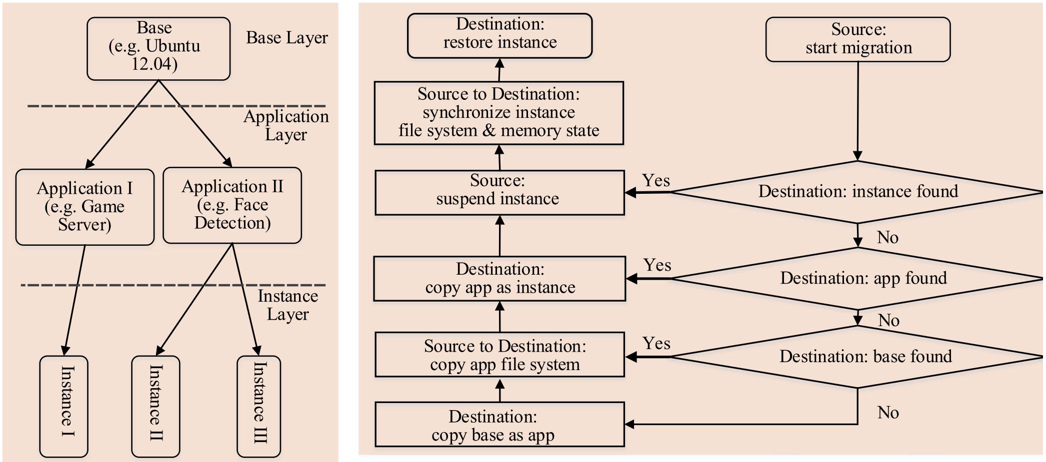


FIGURE 3. **The three-layer framework (left side) and the flow chart of the service migration process (right side).**

stations [27], [37]. As the size of signal messages and state data is very small, the time cost for data transferring only accounts for a tiny part of the whole time of handover process of cellular networks. While for service migration in MEC, the data that should be transferred (e.g. memory state data, application image data, input dataset, etc. [12], [26], [39]) is always very large (e.g., in megabyte or gigabytes). Therefore the time cost for data transferring in service migration becomes a critical factor for seamless service migration.

2) In service migration, users can connect to remote edge servers, while handover in cellular networks must happen if a user is no longer in the coverage of the current serving base station. A user can still continue to receive service from the current edge server even if they are no longer directly connected to each other, because mobile user can still exchange data with remote edge servers with the help of its direct connection edge server as a relay node. Hence, the ongoing service can be placed on any feasible edge server, which gives the service migration problem larger scope [40].
3) In service migration in MEC, between the start edge server and the destination edge server, there may exist various network topology (e.g. remote clouds or other edge servers as intermediate nodes) and communication systems (e.g. Wi-Fi, LTE-U, 4G and 5G) [41]–[43], leading to different network connections and transmission paths for data transmission between them (various transferring latency and process cost). While handover in cellular networks happens only between two neighboring cellular cells [44]. Therefore, the network environments in service migration are more complex than handover in cellular networks.

As a result, service migration in MEC is a problem different from handover in cellular networks, therefore, the handover technologie in cellular networks cannot be directly applied to the problem of service migration.

From these comparison above, we can conclude that service migration can integrate advantages of live migration in data centers and handover in cellular networks and do some adjustments to better adapt to the MEC environment, e.g., large data volume, complex network condition, etc.

III. TECHNIQUES OF MIGRATING RUNNING SERVICE

In this section, we detail the techniques for migrating running services, including a three-layer framework augmented service migration flow and optimization of data transmission. The optimization of data transferring only deals with low level processing in service migration, while the three-layer framework augmented service migration flow improves performance from a higher level view. Here we put them together to give a more comprehensive introduction of the techniques of migrating running service.

A. THREE-LAYER FRAMEWORK AUGMENTED SERVICE MIGRATION FLOW

As shown in Fig. 3 [12], the three-layer framework for migrating running applications is used to optimize the downtime and the total migration time, which divides the service running on edge server into three layers as follows [26]:

1) Base. It includes the guest operating system, kernel, etc., however, no service applications are installed and it can be largely reused by different applications. A copy of this base layer may be stored on most edge servers, so it is unnecessary to be transferred during each migration process.
2) Application. It is a release version of an application with only application-specific data. Like the base, application is unnecessary to be transferred every time, neither, because edge server can download various applications from application stores or official application web sites by itself.
3) Instance. It is the running state of an application, such as CPU, register, non-pageable memory, etc.

The migration process benefits from the above three-layer framework. The whole process of migration is as shown the flow chart of Fig. 3. It should check whether the destination edge server has the copy of the needed base, application to avoid unnecessary data transferring. If the instance can be found in destination edge server, it means that application layer and base layer have already existed there and it is not necessary to copy these two layers from the source edge server. Similarly, if the application can be found in destination edge server, it implies that the base layer has existed there. When migrating a service instance, inspired by pre-copy memory migration, all the memory data is transferred from the source edge server to the destination edge server while the service instance is still running, until pre-fixed criteria is met. Then the running service is suspended and the remaining data is transferred to the destination edge server. At the destination edge server, the service can be reconstructed from a collection of the base, application, and instance data. In this way, we can transfer most of the service data before suspending the service, and service downtime is minimized as much as possible. As the base layer or application layer always has a large amount of data compared to the instance layer (e.g., base package may only have data of hundreds of megabytes or several gigabytes for LXC[1] and KVM,[2] respectively), the three-layer framework helps minimize the transmission time remarkably in the process of migration.

B. DATA TRANSFERRING OPTIMIZATION

Different from the three-layer framework augmented service migration flow in last section, the data transmission process can be further optimized from the following perspectives: [12], [45].

1) REDUCING DATA SIZE

Since network bandwidth is in general the bottleneck of service migration, the amount of data is aggressively reduced to ease the burden of transferred data across the network. As is shown in Fig. 4 [12], reducing the amount of data involves changes tracking, delta-encode, deduplication and compression before it contacts with the network interface.

- **Tracking of changes.** It includes two aspects, i.e., disk tracking and memory tracking. 1) For disk tracking, at the beginning, the system can snapshot all disk data that differ from the corresponding base layer. Then any further disk changes will be logged for subsequent data transferring, and the service can continue to run at the same time; 2) For memory, it is different from the disk tracking, as it would will lead to more overhead on memory write. Memory snapshot is based on a live migration scheme [32], and this process will be iterated several times, sending memory blocks that are changed in the previous iteration period.
- **Delta encoding of modification.** For each changed data block, a delta algorithm is utilized to encode and send out the difference between the data block and the corresponding one in the base layer [32]. The reason is that very small changes are large probability events, and there may exist considerable overlap between the running service and the application.
- **Deduplication.** Deduplication works very well in reducing redundant data. The same parts are removed out at this stage, and they are replaced with pointers to the corresponding blocks [32].
- **Compression.** At this stage, data attempts to be further compressed by using several off-the-shelf compression algorithms (e.g, GZIP, BZIP2 and LZMA, etc.), which vary in compression ratio and processing speed. Multiple instances of the compression algorithms can run in parallel to alleviate CPU-intensive overhead [32].

It is worth noting that the processing cost in the pipeline may lead to CPU bottleneck, rather than data transferring across network. To get rid of this issue, different algorithms and parameter configurations can be applied to make a trade-off between the processing demands and data volume to be transferred.

2) PIPELINED STAGES

As is mentioned above, the execution of the processing stages is pipelined, so they can be processed simultaneously, which can lead to two advantages as follows: 1) downstream stage can be started before the previous stage is completed. For example, data can be transferred via network in parallel to these processing stages; 2) less memory capacity is needed to buffer the temporary data generated by a single stage, as the data is taken away by downstream ones immediately.

3) DYNAMIC ADAPTION

A fixed setting of parameters in above-mentioned stages is difficult to minimize time of the service migration. The reasons are as follows: 1) the relative parameters rely heavily on the transferred data, and can not be known in advance; 2) network bandwidth can change rapidly over a small period of time, and so does for the available processing resources.

Alternatively, service migration performance can be monitored continuously, and the tracked information can be utilized to adapt the processing stage setting to dynamically optimize migration time. More specifically,

- **Throughput calculation of pipeline.** The pipelined system has two potential bottlenecks: 1) processing: if data volume is too large or difficult to process, and aggressive data reduction takes much more time and resources; 2) transmission: if processing stage is not enough to make the data small enough, so network bandwidth encounters problem.

  With respect to those two potential bottlenecks, the throughput of the pipeline system can be obtained as follows. Suppose that the processing sequence in

[1] LXC is a user interface for Linux kernel container. Using a set of powerful APIs and tools, it helps users create and manage containers with ease.

[2] KVM is a virtualization scheme for Linux on X86 hardware virtualization extensions.

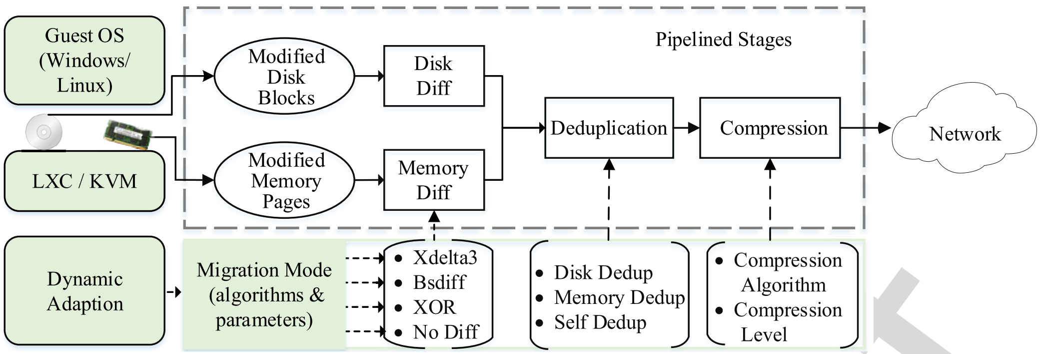


FIGURE 4. **Data transferring optimization in service migration (Note: dedup and diff are respectively short for deduplication and difference).**

pipeline is composed of $n(n = 1, 2, 3, \ldots)$ sequential stages, and each of them consumes input data and generates a smaller version. With a specific set of selected algorithms and parameters (i.e., the *Migration Mode* in Fig. 4), at stage $i(i = 1, 2, 3, \ldots, n)$ we define as follows:

$$p_i = processing\ time,$$
$$r_i = \frac{output\ size}{input\ size}. \tag{1}$$

The processing throughput and network transmission throughput can be derived from processing time and network transmission time as follows:

$$thru_{processing} = \frac{1}{\sum_{i=1}^{n} p_i},$$
$$thru_{network} = \frac{network\ bandwidth}{\prod_{i=1}^{n} r_i}. \tag{2}$$

Since the pipeline overlaps processing and network transmission, the total throughput is

$$thru_{system} = min\{thru_{processing}, thru_{network}\}. \tag{3}$$

Intuitively, it reveals that whether processing or network transmission is the bottleneck.

- **Heuristic adaptation.** Based on the throughput of pipeline above, the migration mode can be selected to maximize the system throughput $thru_{system}$. We write down the $P = \{p_i|i = 1 \sim n\}$ and $R = \{r_i|i = 1 \sim n\}$ to compute various parameter setting. However, they are heavily depending on the actual content (e.g, text, audio, video, etc.) to be transferred. As a result, P and R may generate high misleading result. It has been noted that the trends of P and R are similar in different scenarios, and the ratios for different workloads are obviously different. Although one workload may be quite different another, it influences different algorithms to a similar degree, and the relative performance remains unchanged. Alg. 1 shows an example to determine which operating mode is likely to minimize handoff time. It uses ratios of P (or R) from the real data, i.e., relative values rather than the absolute values. It can adapt to changes of network bandwidth, available processing resources and compressibility of virtual machine modifications.

Algorithm 1 The Heuristic Algorithm to Dynamically Adapt the Migration Mode

1: Measure current P ($P_{current}$) and R ($R_{current}$) values of the running service of current migration mode ($M_{current}$). Measure current network bandwidth by tracking the rate of data block acknowledgments from migration destination;
2: Find P ($P_{profile}$) and R ($R_{profile}$) values of the matching migration mode M. Compute the scaling factor for P and R as follows: $scale_P = \frac{P_{current}}{P_{profile}}$, $scale_R = \frac{R_{current}}{R_{profile}}$;
3: Using these scaling values to adjust P, R values for workload at present. For each migration mode, calculate processing throughput ($thru_{processing}$) and network transmission throughput ($thru_{network}$);
4: Select a migration mode that maximizes the system throughput.

---

4) WORKLOAD DISTRIBUTION

The relative loads on the network and processing change with the ratio of modified and unmodified data blocks on the pipeline system. In fact, the modifications of memory are always non-uniform and highly clustered, which can result in a highly bursty workload on the processing pipeline. This problem comes in two ways: 1) long sequences of unmodified data block transfer the high processing burden to the later stage, which makes the whole processing choked and leave nothing to the network in a very long period of time; 2) at the opposite extreme, long sequences of modified pages may bring about high processing burdens, which require more compression to maintain the full use of the network. Note that change tracking mechanism can only ensure that the modified disk blocks are delivered to the processing pipeline. However, for the memory image, the entire snapshot, including both modified and unmodified pages are processed. As a result,

TABLE 2. Works on strategies for service migration.

Strategies		Advantages	Disadvantages	References
Follow Me Cloud prototype		general framework, can cover most use cases	cannot control the bottom implementation, and optimize well for a specific case	[4], [39], [46], [47]
MDP based service migration	One-dimensional MDP	Easy to use, low computational complexity	as a highly abstract model, not work well in real uses with many parameters	[4], [47], [48]
	Two-dimensional MDP	higher computational complexity than one-dimensional MDP	a more general model than one-dimensional MDP, and can treat more use cases	[4], [39], [49]
Time window based service migration		more general than MDP based models	too many parameters to control	[25], [40], [50]

unmodified data blocks should also be transferred to the destination server, incurring processing if the changes are tracked. When the network is fully used, the best performance can be achieved in network throughput capacity. When the network throughput capacity is small, the data can be compressed and transferred to make it smaller than before.

To solve this problem, workload distribution is employed to balance the workloads during the process of service migration. Specifically, 1) workload distribution randomizes the order of pages on the pipeline system, neither processing nor network resources are idle for long time; 2) what's more, the ratio of modified and unmodified pages does not change much all the time. Consequently, workload distribution helps to get rid of the peak workloads and helps the pipeline system efficiently utilize network and CPU resources.

5) ITERATIVE TRANSFER FOR LIVENESS

As is mentioned above, service migration makes a tradeoff between service downtime and duration of service degradation: 1) if the total migration time is the only one concerned, the post-copy approach contains suspending, transferring, then restarting the service would be the best choice. However, this may break down the running service QoS for long time; 2) the other extreme may unacceptably extend the duration of degraded service.

To solve this problem, inspired by iterative transfer concept from live migration, use it in quite different environments of adaptive service migration state transferring. Unlike live migration, which focuses solely on the volume of data transfer, service migration is sensitive to multiple factors: data volume, processing speed, compression ratio and bandwidth information. It makes use of an input queue threshold to start another iteration and the duration of the iteration to track and log all elements related to the migration speed. If the iteration duration is short enough, the system suspends the service migration and completes the migration operation.

IV. STRATEGIES FOR SERVICE MIGRATION

Here, we review the existing strategies for service migration proposed in recent years. First, we introduce the follow me cloud prototype, which is aimed at seamless migration of ongoing service between a data center and another optimal data center. Then we present the Markov Decision Process (MDP) based service migration strategies, including one-dimensional MDP (i.e., mobile users move along a straight line, e.g., the car on the road) and two-dimensional MDP model (it's a more general case than one-dimensional MDP model, where mobile users move in an area, e.g., in a square). At last, we detail the time window based service migration strategy. Table 2 summarizes three parts of this section.

A. FOLLOW ME CLOUD PROTOTYPE

The FMC allows services to move across federated data centers (DCs), which to some extent can be considered as edge servers. As a user moves, the ongoing service hosted on the current edge server will be migrated once to an optimal edge server. The detailed evaluation criterion for optimality is related to the policy of operators, which is typically based on geographical distance or workload. The cost of service migration is incurred by signaling messages and data transferred between edge servers, and service migration improves QoS of mobile users at the same time. As a result, the migration policy should strike a balance between the incurred cost and QoS improvement induced by service migration [4], [39], [46], [47].

A representative network architecture of FMC concept is as shown in Fig. 5 [39]. The figure shows two main components of FMC, namely FMC controller and edge server/gate way (i.e., DC/GW) mapping entity, that can be considered as two

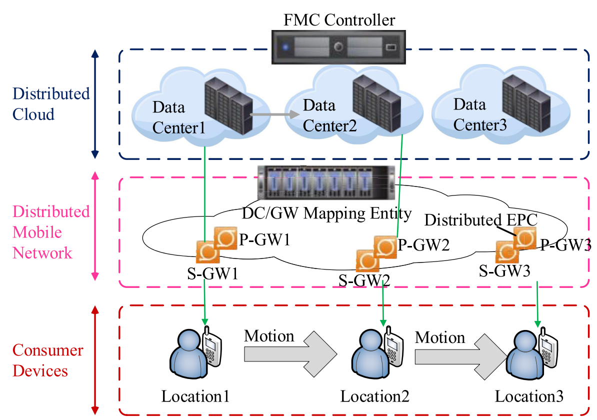


FIGURE 5. Follow Me Cloud prototype.

independent function entities collocated with existing components of mobile cloud computing, e.g., DCs, P-GWs (packet gate way) and S-GWs (service gate way). In the above FMC network architecture, both edge servers and mobile operator network are geographically distributed. Each edge server is mapped to a collection of P-GWs and S-GWs based on their locations. The topology information and location information can be communicated between FMC providers and mobile network operators. FMC controller is to manage and schedule the distributed edge servers.

Service migration demand can be easily observed when one mobile device alters its IP address as a mobile user moves around. This change of information can be certainly observed by the corresponding edge server. A choice on whether to migrate the corresponding ongoing service on the edge server has to be made by the mobile device or the current edge server. This service migration decision should be based on several factors, including but not limited to service type (e.g., a video play with high QoS demand tends to be migrated), data size (e.g., enjoying a movie nearing to its end on your mobile devices, and it should not to begin service migration), etc.

As long as it is decided to migrate the service, the edge server may require the FMC controller to choose a most suitable edge server to start the service migration process. An estimate of the potential cost incurred should be compared against the resource utilization improvement of MEC community and QoS improvement from the point of end users.

Service migration process in FMC architecture can be further modeled using MDP. MDP based service migration method takes into account both the cost and benefit of service migration, and it helps to produce the best policy to decide whether to migrate a service or not. In what follows, the details will be provided.

B. MDP BASED SERVICE MIGRATION

In this section, we present the MDP based service migration strategy, including one-dimensional MDP and two-dimensional MDP.

1) ONE-DIMENSIONAL MDP

One-dimensional MDP is first proposed in [47] and [48], where mobile users are considered to move down a straight line, e.g., the car on the road.

As is mentioned in the former sections, a good service migration model should take into account the balance between cost reduction and high QoS of mobile users. To strike this balance, the service migration decision is modeled as a MDP. Given the distance from a mobile user to the current edge server, MDP based model can decide whether to migrate the ongoing service to an optimal edge server or not. The MDP solution can be implemented inside the FMC controller in the last subsection. To build up the service migration decision model, works in [4] and [47] proposed one dimensional MDP based model, which it considers the distances between mobile users and edge servers as the states, and associates with an action that means whether migrate or not, and defines the corresponding transition probabilities between two states with a definite action and the rewards. In this way, one MDP based model is proposed to solve service migration problem.

Let s_t be a state at time t and $S = \{s\}$ denote the state space that contains all states. In the one dimension (1-D) mobility model, a mobile user has only two possible destinations, namely moving to another edge server with large distance with a probability $0 \leq p \leq 1$, or returning back to the current edge server with a probability $(1-p)$. The state space S is defined as $S = \{0, 1, \cdots, g\}$. Here, $0, 1, \cdots, g$ stands for the possible set of the discrete distances between mobile users and the connected edge servers, and value g means the maximum distance where the service must be migrated to the optimal edge server. Then we introduce the concept of action set. For example, $A_s = (a_1, a_2)$ can denote the action set available at state s, where action a_1 means that the service is migrated to an optimal edge server, while action a_2 means that mobile devices are still served by the same edge server. For a given action a, there will be a state transition from state s to another state s', with which there is also a reward $r(s, s', a)$. Fig. 6 [51] illustrates one dimensional MDP model that can be integrated into FMC architecture, where FMC controller observes the current state s of mobile users in the network and associates a set of possible actions A_s to it. When the service migration is triggered, it always means that they have been at another edge server, so the state is always 0 after service migration.

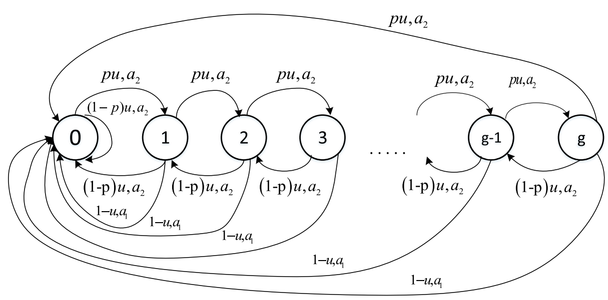


FIGURE 6. One-dimensional MDP based service migration. Action a_2 means that mobile devices are still served by the same edge server, a_1 means that the service is migrated to an optimal edge server. Value g means the maximum distance where the service must be migrated to the optimal edge server. Value μ can be considered as the probability that user moves.

Without loss of generality, A_s means a unique action set at state s. Then we define the transition matrix Q, in which $q(s|s')$ denotes the transition rate from state s' to s. Service migration policy associates an action to each state. That is to say, policy can be considered as a function of the state, where it takes a state as input, and gives an action as output. As a result, whether migrating a service or not is totally decided by the actual state. It is worth noting that the state space is finite, i.e., $0, 1, \cdots, g$. The reason is that in our settings, after a certain distance (g) from the current edge server, the service must be automatically migrated to the optimal edge server in case of service interruption.

To get the one dimensional MDP model, we should normalize the above transition probabilities by the following. According to MDP theory, if the values of transition rate in matrix Q are all bounded, the stay times in all states are exponential with $t(s|s,a)$. Then there exists:

$$sup_{(s\in S, a\in A_s)}[1-p(s|s,a)]t(s|s,a) \leq c < \infty, \tag{4}$$

where $p(s|s,a)$ denotes the probability of staying in the same state after taking action a at state s, and c is a constant value. After that, we define an equivalent normalized process with state-independent exponential stay times using parameter c and transition probabilities:

$$p(s'|s,a) = \begin{cases} 1-\dfrac{\left(1-q(s'|s)\right)t(s'|s,a)}{c} & s=s' \\ \dfrac{q(s'|s)t(s'|s,a)}{c} & s \neq s'. \end{cases} \tag{5}$$

Suppose that the stay time of one mobile user in a state follows an exponential distribution with mean $1/(\mu-1)$. Then by setting $c=\mu-1$, the transition probabilities are defined by the following:

$$p(s'|s,a) = \begin{cases} 1 & s'=0,\ a=a_1 \\ p & s'=s+1,\ s\neq g,\ a=a_2 \\ 1-p & s'=s-1,\ s\neq 0,\ a=a_2 \\ 0 & else. \end{cases} \tag{6}$$

Note that when in state $s=g$, the only available action is a_1, which means that when the mobile device moves to another edge server where the distance is larger than the maximum g, the service migration action is automatically triggered.

For $t\in N$, let s_t, a_t and r_t denote state, action and reward at time t, respectively. Let $P^a_{(s,s')}=p[s_{t+1}=s'|s_t=s, s_{t+1}=s', a_t=a]$ denote the transition probabilities and $R^a_{(s,s')}=E[r_{t+1}|s_t=s, s_{t+1}=s', a_t=a]$ denote the expected reward. A policy π is a mapping between a state and an action, and can be denoted as $a_t=\pi(s_t)$. In the process of service migration, reward is a function of the cost of migrating one service and the quality obtained from the new state. Given a discount factor $0\leq\gamma\leq 1$ and an initial state s, the total discount reward policy $\pi=(\theta_1,\theta_2,\theta_3,\ldots,\theta_N)$ can be denoted as follows:

$$v^\pi_\gamma = E^\pi_\gamma\{\sum_{t=1}^{\infty}\gamma^{t-1}r_t\}. \tag{7}$$

Reward function $r(s',s,a)$ explicitly depends on the transitions among states. According to [52], the normalized reward function $R(s',s,a)$ is written as follows:

$$R(s',s,a) = r(s',s,a)\frac{\alpha+\beta(s',s,a)}{\alpha+c}, \tag{8}$$

where $\beta(s',s,a)$ is the transition rate between state s and s' when taking action a, and α is a predetermined constant. Let $v^*(s)$ denote the maximum discounted total reward, i.e., $v(s)=max_{\pi\in\Pi}v(s)$, given the initial state s. Using the predefined denotations, we can formulate $v^*(s)$ by the following:

$$v^*(s) = max_{\pi\in\Pi}\{R(s',s,a)+\sum_{s'\in S}\gamma P[s'|s,a]v(s')\}. \tag{9}$$

The optimal solution of Eq. (9) includes $v^*(s)$ and $\pi^*(s)$. In the area of service migration, the optimal policy $\pi^*(s)$ indicates the decision as to which network and which data center the mobile user should migrated to with each state.

2) TWO-DIMENSIONAL MDP

Two-dimensional MDP model is first proposed in [4] and [39], which is a more general case than one-dimensional MDP model, where mobile users move in a 2D area, e.g., in a square.

Typically, a cellular network is considered to be composed of multiple adjacent hexagonal cells (Fig. 7a). User mobility can be considered as a random walk model, whereby mobile users come into the six adjacent cells with the same probability (Fig. 7a), i.e., $p=1/6$. Fig. 7 [39] shows a cellular network with $K=5$ rings of cells. The service migration is triggered as the mobile device is equal to or large than K hops away from the current edge server. Here, the distance means the number of hops from the location of mobile user to the current edge server. So we obtain a Markov chain with state space $\{C_{(m,n)}|0\leq m\leq(K-1), 1\leq n\leq 6m\}$, which, however, suffers from state space explosion problem when K value is high. However, according to works in [49] and [53], we can reduce the state space by aggregating states with the same behavior. Then we can obtain a new chain with less number of states.

We give an example to show the state aggregation process. In Fig. 7a, it can be seen that mobile users in the first ring have the same behavior and can move to each neighboring cell with the same probability. That is, mobile devices come back to the cell with the optimal edge server with probability p, stay in the same ring (i.e., the same distance from the optimal edge server) with probability $2p$, and move to second ring with probability $3p$ [39]. As a result, all states of the first ring can be aggregated into one state. As to the second ring, we differentiate it into two cases. The mobile device leaves the service area with probability $3p$ in the first case, instead of $2p$ in the second case. Therefore, we choose the concept of aggregated states in the two-dimensional service migration, instead of the initial states. For example, one aggregated is state $C^*_{2,0}$, which aggregates states $\{C_{2,1}, C_{2,3}, C_{2,5}, C_{2,7}, C_{2,9}, C_{2,11}\}$, another is $C^*_{2,1}$, which aggregates states $\{C_{2,2}, C_{2,4}, C_{2,6}, C_{2,8}, C_{2,10}, C_{2,12}\}$. Using this method, we can obtain a chain with less states in Fig. 7b, which shows the transition diagram of the aggregated Markov chain for the service migration when the mobile device is K hops away from the optimal edge server. We can derive the steady state probability of the aggregated states C_m and C^m_m, respectively. The functions of these steady state probabilities

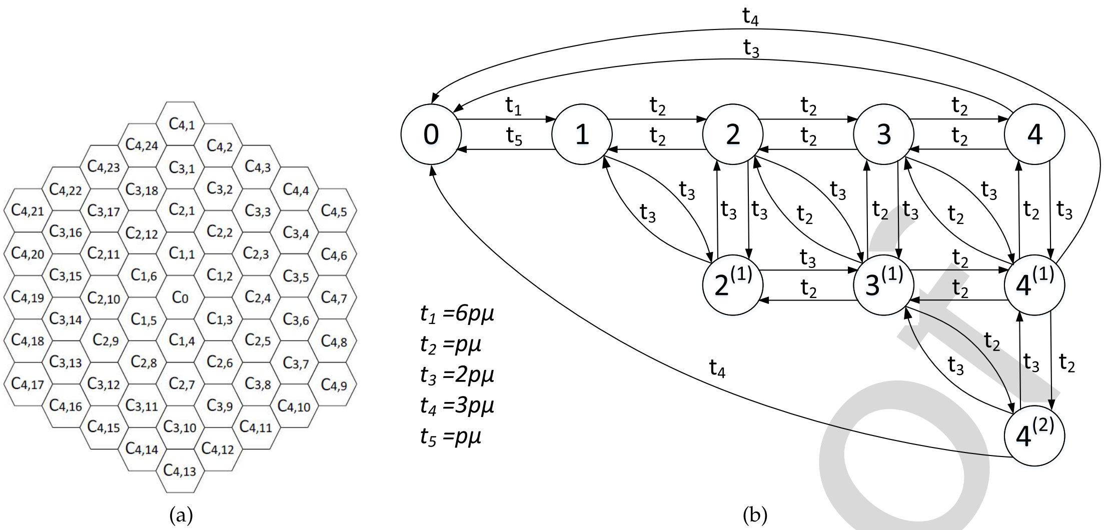


FIGURE 7. **Two-dimensional MDP based service migration. The Markov chain here is more complex than that in Fig. 6, which is a one-dimensional MDP model. The same to Fig. 6, value μ is considered as the probability that user moves. At state 0, if user moves, the next state is definitely 1. So the probability is $\mu = 6p\mu = t_1$. In this way, we can get other transition probabilities in this figure. (a) A typical cellular network on two dimensional plane. (b) Markov chain in case of $K = 5$.**

are as follows:

$$\begin{cases} \pi_0 = \dfrac{1}{6}\pi_1 + \dfrac{1}{2}\pi_{K-1} + \dfrac{1}{3}\sum_{n=1}^{\lceil \frac{K-2}{2} \rceil} \pi_{K-1}^{(n)} \\ \pi_1 = \pi_0 + \dfrac{1}{3}\pi_1 + \dfrac{1}{6}\pi_2 + \dfrac{1}{3}\pi_2^{(1)} \\ \pi_2 = \dfrac{1}{6}\pi_1 + \dfrac{1}{6}\pi_3 + \dfrac{1}{3}\pi_2^{(1)} + \dfrac{1}{6}\pi_3^{(1)} \\ \pi_{K-1} = \dfrac{1}{6}\pi_{K-2} + \dfrac{1}{6}\pi_{K-1}^{(1)}, \qquad \forall 3 \le m \le K-2 \\ \pi_m = \dfrac{1}{6}\pi_{m-1} + \dfrac{1}{6}\pi_{m+1} + \dfrac{1}{6}\pi_{m-1}^{(1)} + \dfrac{1}{6}\pi_{m+1}^{(1)}, \end{cases} \tag{10}$$

where $\lceil x \rceil$ denotes the smallest integer larger than or equal to x. We have

$$\begin{cases} \pi_2^{(1)} = \frac{1}{3}\pi_1 + \frac{1}{3}\pi_2 + \frac{1}{6}\pi_3^{(1)} \\ \pi_3^{(1)} = \frac{1}{3}\pi_2 + \frac{1}{3}\pi_3 + \frac{1}{3}\pi_2^{(1)} + \frac{1}{6}\pi_3^{(1)} + \frac{1}{6}\pi_4^{(1)} + \frac{1}{3}\pi_4^{(2)} \\ \pi_4^{(1)} = \frac{1}{3}\pi_3 + \frac{1}{3}\pi_4 + \frac{1}{6}\pi_3^{(1)} + \frac{1}{6}\pi_5^{(1)} + \frac{1}{3}\pi_4^{(2)} + \frac{1}{6}\pi_5^{(2)} \\ \qquad \forall 5 < m < K-1 \\ \pi_m^{(1)} = \frac{1}{3}\pi_{m-1} + \frac{1}{3}\pi_m + \frac{1}{6}\pi_{m-1}^{(1)} + \frac{a}{6}\pi_{m+1}^{(1)} \\ \qquad + \frac{1}{6}\pi_m^{(2)} + \frac{a}{6}\pi_{m+1}^{(2)}, \end{cases} \tag{11}$$

where

$$a = \begin{cases} 1 & \text{if } 5 \le m \le K-2 \\ 0 & \text{if } m = K-1. \end{cases} \tag{12}$$

We can also compute the value of $\pi_m^{(n)}$, $\forall 6 < m < K-1 \wedge 2 \le n \le \lceil \frac{m-1}{2} \rceil - 1$, $\pi_{2l+1}^{(l)} \forall 2 \le l \le \frac{K-2}{2}$. That is to say, we can obtain all of the steady state probability of the aggregated states.

With these solutions, we can obtain more attributes of two-dimensional service migration, such as the mean value of the distance, the probability of the optimal edge server connection, cost of service migration, service migration duration, etc. [39].

The concept of FMC prototype is mainly described in Section 4.1, while the MDP based service migration algorithm is mainly described in Section 4.2. They are at different levels in service migration.

C. TIME WINDOW BASED SERVICE MIGRATION

Compared to MDP based service migration above, time window based service migration deals with the problem from another point of view. The goal of time window based service migration is to search the optimal service placement sequence that minimizes the average cost over a given time window [25], [50]. In these works, a look-ahead window is defined as a time period in the future that can be predicted. The model contains two sequential parts: 1) suppose that there exists a method to obtain the prediction error in the future, how to search the optimal window size to minimize the average cost; 2) with a fixed size of time window, how to find the optimal sequence to place the ongoing service.

Compared to MDP based service migration, time window based service migration can deal with a more general setting, such as heterogeneous cost function, network structure and mobility pattern. Cost of service migration may incur in two ways, namely cost in running a service on an edge server and cost in transferring data in a specific migrating procedure. What is more, it supposes that an underlying function can be found out to predict the two kinds of cost in the future time, which includes but is not limited to existing approaches such as [51], [54], and [55]. As to the designed prediction function, the predicted future cost sequence may be different from the actual cost, but it can guarantee the upper bound of the possible deviation. Unlike MDP-based method in [4], [39], and [47], time window based service migration does not need the probability distribution of the cost, which makes it can be applied to more scenarios, where the pattern of users mobility follows a Markov chain model. Time window based service migration takes into account the dynamics of resource availability caused by user mobility, which is quite different from the supposed of static network conditions and fixed resource demands under complicated network topology [56], [57].

We detail the time window based service migration in two parts, i.e., optimal size of the look-ahead window in the future and service placement finding based on prediction cost with optimal look-ahead window size.

1) OPTIMAL SIZE OF THE LOOK-AHEAD WINDOW IN THE FUTURE

This part elaborates how to find the optimal size of the look-ahead window in the future [25], [40], [50].

Suppose that the optimal window size $0 < T \leq T_{max}$, where T_{max} is upper bounded time induced by the service duration. If the future prediction cost function has no deviation from the actual cost, $T = T_{max}$ setting is optimal as it gives the best long-term performance. However, this is impractical for the fact that the farther look-ahead we look in the future, the more uncertainty and deviation about the cost we encounter. That is to say, if the window size T is too large, we will obtain much worse prediction performance and the prediction cost may be far away from the actual cost. The bad performance of prediction cost will generate a very bad solution of the service placement sequence. As a result, we have to find the optimal look-ahead window size that can minimize both the impact of prediction deviation and the impact of dividing the look-ahead time period for optimal window. Window size cannot be accurately set, because it is related to many factors, which is not known before. Thus, if the window size is too large, the prediction is not accurate enough.

For more details of optimal look-ahead window size, please refer to the works [25], [40], [50].

2) SERVICE PLACEMENT FINDING BASED ON PREDICTION COST WITH OPTIMAL LOOK-AHEAD WINDOW SIZE

If we have obtained the the optimal look-ahead window size T, then we can find the optimal placement sequence π_T, according to the following steps as in Alg. 2.

Algorithm 2 Placement Sequence Algorithm

1: Initialize $t_0 = 1$;
2: Let $t_e = min\{t_0 + T - 1, T_{max}\}$. At the beginning of time slot t_0, find $\pi_T(t_0, \cdots, t_e) = \arg\min_{\pi(t_0,\cdots,t_e)} \sum_{t=t_0}^{t_e} D^{t_0}_{\pi(t_0,\cdots,t_e)}(t)$,which $\pi(t_0, \cdots, t_e)$ denotes the placement sequence for time slots $t_0, \cdots, t_e$, and $D^{t_0}_{\pi}(t)$ can be obtained using the prediction cost function;
3: Apply the service placement $\pi_T(t_0, \cdots, t_e)$ in time slots $t_0, \cdots, t_e$;
4: If $t_e < T_{max}$, set $t_0 = t_e + 1$ and go to step 2. If not, stop the algorithm.

Note that in the above service placement algorithm, once the placement in the last window is completely solved, we need make the placement decision in the current time slot. So the vector π_T can be found in real-time, which is fit in the high dynamics of network condition and computing resources in MEC. The value of $D^{t_0}_{\pi(t_0,\cdots,t_e)}(t)$ also depends on the placement in time slot t_0-1. When $t_0 = 1$, $\pi(t_0-1)$ can be regarded as any dummy variable for the fact that the migration cost $w(1, :, :) = 0$. The equation of the placement sequence π_T means that, at the beginning of time slot t_0, it finds the optimal placement sequence that minimizes the prediction cost over the next time slot up to t_e, given the location of the service in previous time slot $t_0 - 1$.

Based on the above assumptions and analysis, the service placement problem can be considered as a shortest-path problem with values of $D^{t_0}_{\pi}(t)$ as weights. Specifically, each edge stands for one possible service placement decision in the corresponding two adjacent time slots and the weight on each edge means the prediction cost for such service placement decision. The placement before time slot t_0 has been found out. We define a dummy node at the end of look-ahead window, which is assigned zero weight to other nodes to ensure a single shortest path to be found. Obviously, the shortest path with minimum sum of weight from node $\pi(t_0 - 1)$ to the defined dummy node can be found with the help of some shortest path algorithms and the nodes on the shortest path give the optimal service migration solution $\pi_T(t_0, \cdots, t_e)$.

V. HOSTING APPLICATION COMPONENTS

An application may consist of several components. Besides, multiple applications can simultaneously use the MEC infrastructure, such as edge servers. Resource isolation (especially, memory) across components of different applications is necessary for the security and integrity of the individual applications; even within an application such isolation between the application components is beneficial from the point of view of bug proliferation and performance tuning. We will explore the pros and cons of full blown *virtual machine technology*, *container technology* and *agent technology*, from the point of view of hosting application components.

A. VIRTUAL MACHINE

Virtual machine is one of enabling technologies for data centers and is the basis for accountability and containment of resource usage. Additionally, live migration of virtual machine has been extensively investigated to enable load balancing and resource provisioning in data centers [73]. More recently, VMWare [31] and Xen [74] have implemented live migration of virtual machines with downtime ranging from tens of milliseconds to seconds. As live migration of virtual machine is a mature technology used in data centers of cloud computing, many existing works on service

TABLE 3. Hosting application for service migration.

Host	Disadvantages	Advantages	References
Virtual Machine	slow boot and running, large data volume to store and transfer	high isolability and security	[4], [12], [29], [39], [46], [47], [58], [59], [60], [61], [62], [63]
Container	bad cross-platform performance, e.g., a container on Windows won't work when transferred to Ubuntu	less data, high starting speed	[25], [26], [62], [63], [64]
Agent	preliminary stage, and no existing framework to use directly	administrative convenience, small data to transfer, rapid boot and running	[41], [65], [66], [67], [68], [69], [70], [71], [72]

migration in MEC take virtual machine as the host for application components [4], [12], [26], [39], [61], [63], [75]–[78]. Ha *et al.* [12] discuss the limitations of live virtual machine migration for use on edge devices, examine the impact of user mobility on cloudlet offload, demonstrate that even the most general user mobility can bring about considerable network degradation, and propose a *VM handoff* technique for seamlessly transferring a runtime virtual machine instance to a better offload site as users move. To reduce the downtime during service migration, Machen *et al.* [26] propose a layered framework to transfer ongoing applications that are hosted in virtual machines, which does not need users to have extensive knowledge on the technical details of service migration. Taleb *et al.* [4], [39] applies a MDP based algorithm to cost-effective, performance-optimized service migration decisions, and two alternative schemes to ensure service continuity and disruption-free operation in the context of FMC, which is tailored to an interoperating decentralized mobile network/federated cloud architecture. In this work, they mainly consider two types of time that affect the service continuity, i.e., the time required for transforming a virtual machine to another type (particularly if two relative edge servers are using different hypervisors), and the time required for service data transferring. Refaat *et al.* [61] propose a service migration solution to select the best destination in service migration in VANET, which aims to perform efficiently in dealing with rapid dynamics of data center topology with minimum *roadside unit* intervention. Virtual machine technology is also applied in MEC for service deployment and the migration of location-aware services [63]. Satyanarayanan *et al.* [75] propose the concept of cloudlet to exploit standard virtual machine technology in MEC. Yao *et al.* [76] present the roadside vehicular cloud architecture in Vehicular Ad-Hoc Networks (VANET) using cloudlet, and study how to migrate the virtual machines as vehicles move to reduce transferring cost. Recently, many works propose approaches to virtual machine migration with less involvement of the hypervisor [77] or with a reduction in the startup time by using delta encoding between an original virtual machine instance and the changes that occurred during execution [78].

However, despite such advances in virtual machine migration techniques, given the latency requirements of situation awareness applications, full blown virtualization may be impractical for hosting application components in the MEC environment.

B. CONTAINER

Container based service migration is a relatively new area and it needs to be studied systematically. In comparison to virtual machines, containers are much more efficient for creating service bundles for one cloud to another transferring [79], [80]. Here, containers are preferred than virtual machines because they share more platform resources in common, whereas, a virtual machine tends to hold most resources in migrating services [64], so a container is always much smaller than a virtual machine. As edge servers in MEC have limited bandwidth, unstable network connectivity, storage and processing capability, running container-based applications on them will benefit much more in migrating services.

More specifically, containers have the following advantages to support service migration in MEC:

- Complexity can be reduced through container abstractions. Containers avoid reliance on low-level infrastructure services, which decreases the complexity of dealing with those platforms.
- Automation can be supported with containers to maximize the portability. Through automation, tasks can be conducted without much manual efforts, such as migrating containers among edge servers.
- Better security and governance can be achieved by placing services outside, rather than inside, the containers. In many cases, security services are platform-specific instead of being application-specific, which helps to provide better portability and less complexity in implementing and operating.
- Higher computing capability can be provisioned as a service can be split into many separate containers. These containers can run on different physical machines or edge servers to obtain better performance.
- In the container technology, applications contained in the containers share the OS. Consequently, the memory footprint of containers is significantly smaller than in a hypervisor environment, allowing hundreds of containers to be hosted on a physical host. Since the containers use the host OS as a base for system services, restarting a container (upon container migration) does not necessarily restart the OS.
- Once a container is installed, only the extra different layers, such as additional binaries and libraries, need to be migrated to correctly execute the handlers in the context of edge server.

Given the above-mentioned advantages, more and more mainstream operating systems begin to adopt container technology to provide isolation and resource control, which has demonstrated great potential for service migration. Mirkin *et al.* [81] propose saving the complete state of a container (i.e. checkpointing), transferring it to another host, and restarting it as implementing in OpenVZ.[3] A container allows users to checkpoint the running state of a container and restart it later on the same or a different host, which is transparent for ongoing services and network conditions. OpenVZ is based on CRIU,[4] which is a project to implement checkpoint/restore functionality for Linux. In 2016, live migration of container was also realized using CRIU.[5] Especially in recent years, Docker as a standard for Linux containers [80], has been adopted extremely successfully by Google, IBM/Softlayer, and Joyent in public cloud platforms [79]. In this context, Machen *et al.* [26] proposed to use containers in their service migration framework, and showed that containers perform favorably than virtual machines. Apart from that, Wang and Serral-Gracià [25], Montero *et al.* [62] and Saurez *et al.* [63] also take into account container when performing service migration in MEC.

C. AGENT

In computer science, an agent is a computer program block that performs tasks in a relationship of agency with other entities [82], [83]. An agent has the following characteristics [84]: 1) autonomous: it runs without human interventions, and can control its external behaviors and internal states by itself; 2) social: it can sense, process and react to humans or other agents to perform better; 3) reactive: it perceives the change of environment and responds in turn in time; 4) proactive: its behaviors to the environment are highly goal-directed; 5) mobile: it is able to travel between different hosts in a network; 6) truthful: it will not deliberately output false information; 7) benevolent: it always tries to perform what is asked; 8) rational: it performs in order to achieve its goal, not the other way around; 9) learning: it can learn to fit the environment better to be stronger with time.

As an agent has the above-mentioned advantages, service migration based on agents will impose less requirements on edge servers other than providing run-time environment, and it releases the management burden of edge servers and mobile terminals using autonomous agent-based application partition [72]. While in service migration process hosting of virtual machines and containers, these management burden relies heavily on the support from the underlying virtualization technology [77]. Compared to service migration with virtual machines and containers, service migration with agents can perform better in the dynamic and heterogeneous environment in MEC (e.g. hosts of virtual machine, containers, and even physical machines; and rapidly changing network conditions, etc.). For example, an agent implemented in JADE[6] can be migrated among virtual machines, containers and physical machines, as long as they are equipped with Java runtime environment [84].

With these advantages, agent technology has been widely applied in cloud computing, MEC and micro grids [41], [65]–[72], which shows great potential. Angin and Bhargava [65] propose a framework based on agent in mobile cloud computing, and show that application encapsulation based on agent is particularly useful due to the capability of moving without the intervention of the caller and self-cloning. These results can be applied in MEC, which has many common characteristics with mobile cloud computing. Then Angin *et al.* [66] propose to make use of autonomous agents to offload dynamic computation in MEC. As to the security issue of mobile cloud computing, Angin *et al.* [67] also propose a mobile cloud computing model based on agent to deal with code tampering, where agents are integrated with integrity verification functions. Kumar *et al.* [68] propose mobile agents to alleviate the issue of unstable and intermittent wireless network connectivity and low bandwidth in wireless/mobile network. Alami-kamouri *et al.* [69] survey mobile agent technology in fields of mobile computing, network management and telecommunication, security issue, etc., in a flexible way by using interaction with other agents on the network. Luo *et al.* [70] propose a multiple agent framework to promote energy sharing among the massively distributed autonomous micro grids, which is similar to MEC environment and relieve the energy imbalance problem by forming micro grid coalition with agents. Zhu *et al.* [71] apply the agent technology in cloud computing environment to design an agent-based scheduling mechanism to deploy real-time tasks and dynamic resources. Fareh et al [72] propose that autonomous agents can make the clouds smarter in their interactions with users and more efficient in resources allocation. Gani *et al.* [41] summarize the application of agent technology in the field of the interworking for seamless connectivity.

The pros and cons of *virtual machine technology*, *container technology* and *agent technology* are summarized in Table 3. Compared to virtual machine and container technologies, agent has advantages of administrative convenience, small data to transfer, rapid boot and running, etc., which is quite suitable in IoT environment. However, agent technology in mobile edge computing is at its preliminary stage, and there are no existing frameworks to use directly. As a result, much work should be done to develop an agent tool to apply agent technology into IoT applications.

[3] A container-based virtualization for Linux. OpenVZ can create many different isolated containers on a single physical edge server, which enables better server utilization and ensure that different services do not conflict with each other.

[4] http://criu.org

[5] http://rhelblog.redhat.com/2016/12/08/container-live-migration-using-runc-and-criu/

[6] JADE is short for JAVA Agent DEvelopment Framework, which is a software framework to develop agent applications.

VI. ISSUES AND CHALLENGES

In this section, we identify and discuss some research challenges in service migration in MEC, including design of QoS-aware edge server selection algorithm, selection algorithm of migration path with both of latency and cost, and virtual resource allocation strategy on edge servers, and development of a high service migration mechanism to ensure service continuity.

A. QOS-AWARE EDGE SERVER SELECTION ALGORITHM

For smooth service migration in MEC, an efficient edge server selection algorithm is needed to select the optimal target edge server. In general, two factors should be taken into account: users' trajectory and QoS utility. On the one hand, existing research works rarely explores users' trajectory data and the prediction of their movement, and adopts a random mobility model instead [85]. However, users' mobility pattern (e.g. direction and velocity) has a significant influence on the construction of the candidate edge server set (e.g. the size of set of candidate edge servers), and the users' trajectory data can be used to predict users' movement. On the other hand, existing literatures pay less attention on the affect of QoS utility (network latency, energy consumption and cost) on the selection of edge servers in service migration, therefore, hardly select the edge server with the highest QoS utility [86]–[88]. Without considering users' trajectory data and QoS utility, the accuracy of edge server selection and the efficiency of service migration decrease.

To develop a QoS-aware algorithm to improve edge server selection, we should overcome the problems such as how to integrate user's trajectory data and QoS utility into the server selection algorithm. The research can be divided into the following parts: firstly, develop user moving model using users' trajectory data to predict user movement, then construct the candidate edge server set; secondly, devise QoS utility function of a given edge server based on QoS indicators (e.g. network latency, energy consumption and cost); at last, based on the designed QoS utility function, select the candidate edge server with the highest QoS utility as the target edge server of the service migration. The key issues are user mobility, QoS utility function design, and the selection algorithm of edge server.

B. SELECTION ALGORITHM OF MIGRATION PATH WITH BOTH OF THE LATENCY AND COST

The related data on the edge server (e.g. the run-time state data of the edge service on hard disk and memory) should be transferred to the selected target edge server in the process of service migration [12], [26]. Between the start edge server and the target edge server, there may exist various network top topology (e.g. remote clouds or other edge servers as intermediate nodes) and communication system (e.g. WiFi, LTE-U, 4G and 5G) [43], which leads to different network connections and transmission paths for data transferring between them (various transferring latency and cost). Therefore, selection algorithm of migration path is essential. Existing work selects the migration path randomly and rarely considers the heterogeneity of network, as well as latency and cost, leading to high service migration expense (e.g. latency and cost) and low transferring efficiency of network (including edge network and core network) [12], [26], [89], [90].

To this end, we can apply network optimization theory and propose a service migration path selection method by taking consideration of both network latency and cost. The main idea is to transform the migration path selection problem with both latency and cost into a multi-objective optimization model, and propose path selection on latency and price in service migration of MEC, and aim to choose the best set of available transferring paths that can minimize the total transferring time with constrictions on bandwidth and price of each network connection for the data transferring in a service migration. Service migration demand can be easily observed when mobile device alters its IP address as mobile user moves around, and the *Service Migration Decision Center* then solves the path selection problem. Note that every network connection has its inherent bandwidth and price attributes, which are relative to the transmission length, access technique, current workload, etc. We analyze this problem from two aspects, i.e., the network operator and mobile user. On the one hand, due to various prices of network connections, network operator should choose the best transferring paths or network connections to save money of providing data transferring service. On the other hand, for mobile users, minimizing transferring time during a service migration can improve QoS/QoE. The best case for transferring time minimization is to realize seamless service migration (i.e., without any disruption to ongoing edge services, a mobile user is able to freely move over a significant geographic area). The basic principle is as follows: firstly, monitor the real-time network condition (e.g. bandwidth, network style information, and distance between two nodes), construct the latency and cost matrix; secondly, based on the proposed expense function, design the optimization model of migration path selection; at last, find the optimal service migration path using mixed integer programming method. The research issues include expense function design, path selection algorithm and parameter optimization.

C. VIRTUAL RESOURCE ALLOCATION STRATEGY ON EDGE SERVERS

The diverse demands of virtual resources (e.g. computation, network and storage resources) of the edge service that be transferred exists in service migration. On the one hand, the run-time state has changed, which leads to different demand of virtual resources. On the other hand, the inherent diversity of edge service (e.g. real-time tasks or batch tasks) results in different demand of virtual resources [66], [67]. The simplest strategy that allocates more resources than the actual needs for each edge service will ensure users' QoS (e.g. low network latency and energy consumption), however, it will lead to low utilization efficiency of edge servers and considerable waste of resources. Meanwhile, this strategy will increase the payment of each subscriber as the payment is positively related with the allocated virtual resources. Existing work has considered the resources at the user end and the network condition, but has not taken into account the virtual resource allocation strategy on edge servers [91]. An extensive resource allocation strategy is often employed (e.g., allocating more resources than that it really needs for each edge service), which will cause high user cost or low users' QoS.

To this end, we can put emphasis on the demand diversity of virtual resources (e.g. computation, network and storage resources) of the edge service, and study the optimal virtual resources optimization allocation strategy. The main idea is to assess the demand for different resources of various edge services. The basic principle is as follows: firstly, based on instruction analysis and the computation time ratio of different modules of the migrated service, design the model to evaluate resource demand; secondly, consider time, energy, cost and other factors, and transform virtual resources allocation problem into a multi-objective optimization model; at last, solve this problem using the improved heuristic algorithm. The key research issues include but not limited to: how to evaluate resource demand given the task to be processed and current resource allocation, how to design the multi-objective optimization model with constraints to take into time, energy, cost as input to solve the virtual resources allocation problem, how to design the above-mentioned utility function, and how to adapt the current heuristic algorithm, such as ant colony algorithm or particle swarm optimization, into MEC environment to allocate virtual resource efficiently.

D. AI BASED STRATEGIES FOR EFFICIENT SERVICE MIGRATION DECISIONS

The mathematical models, such as MDP, are applied to make efficient service migration decisions. Elegant though, mathematical models are based on simple assumptions, thus can not cope with more complex condition and a large number of different parameters [40]. This property restricts the application of simple mathematical model in the field of service migration. Many other factors should be taken into account when making service migration decisions, such as the heterogeneity (many different kinds of hardware) and dynamics (topology and network condition change rapidly) of the edge servers in MEC, real-time requirements when users are moving fast, etc.

Recently, artificial intelligence (AI) technology, represented by deep learning [28], [92] and reinforcement learning [93], [94], is developing very fast, and can help solve this complex problem. AI technology can learn from massive history data, and efficiently react to the dynamic condition. It is necessary to study how to apply AI for making efficient service migration decisions. To apply AI into efficient service migration decisions, we should overcome the following problems, such as data source selection, as there are too many data that can be poured into the AI based method, and many of them may not helpful to our problem. The other is how to design the AI system, such as what algorithm to choose to integrate MEC better into it.

E. BLOCKCHAIN TECHNOLOGY TO SOLVE TRUST ISSUE IN SERVICE MIGRATION

Trust issue can not be neglected in service migration in MEC [66]. Edge servers may belong to different participants, e.g. telecom operators, internet companies, home users, etc. As a result, there is no a centralized administration for different stakeholders and heterogeneous hardwares, thus it is difficult to solve the trust issue in service migration. The environment results in security risk of sending data to trustless edge servers, and this issue is hard to overcome due to large computation burden induced by complex mechanism.

The trust issue in service migration in MEC can be solved by blockchain technology for its good property [95]. A blockchain is a continuously growing list of records, called blocks, which is linked one by one and secured using cryptography [96], [97]. It is inherently resistant to the modification of data. The reason is that once it is recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks and a collusion of the network majority. As a result, a blockchain can serve as an open, distributed operating system that can efficiently record interactions (e.g., transactions) between two individuals or agents and in a verifiable and permanent way; therefore, decentralized consensus can be achieved with a blockchain.

However, distributed Apps on the existing blockchain system (e.g, ethereum) have slow reaction times when it comes to saving information. Simple operations take tens of seconds and occasionally a couple of minutes. It happens when you send a transaction and wait for it to be verified. It is also the case for other distributed technologies. It is not uncommon to wait 30 seconds for pictures from IPFS[7] to save or load. As a result, we should consider the waiting time as a significant problem when we apply blockchain technology into MEC, as users these days are not used to waiting. So the most important problem to be solved is how to minimize the verification time when users make transaction on the blockchain platform. One way is to develop a customized blockchain system with less block generating time for MEC.

VII. CONCLUSION

In this paper, we have reviewed the state-of-the-art literature on service migration in MEC, which ensures service continuity for moving users by migrating the service on the direct connection to remote edge server to the near one with better QoS. We have presented two similar concepts that are closely related to service migration, and compared them for better understanding the features of service migration. In addition, the existing strategies for service migration are categorized and summarized. Moreover, we have discussed the pros and cons of the three hosting technologies for mobile application components. We also have highlighted some research directions and challenges in service migration in MEC, which need further investigation.

[7] https://ipfs.io/

ACKNOWLEDGMENT

The authors would like to thank Yuanzhi Li, Xiaojuan Wei and Xiaoyu Zhang, for discussions and suggestions.

REFERENCES

[1] U. Drolia *et al.*, "The case for mobile edge-clouds," in *Proc. 10th Int. Conf. Ubiquitous Intell. Comput. Auton. Trusted Comput. (UIC/ATC)*, Dec. 2013, pp. 209–215.
[2] Y. Gao, W. Hu, K. Ha, B. Amos, P. Pillai, and M. Satyanarayanan, "Are cloudlets necessary?" School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-139, 2015.
[3] X. Ma, S. Zhang, P. Yang, N. Zhang, C. Lin, and X. Shen, "Cost-efficient resource provisioning in cloud assisted mobile edge computing," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2017, pp. 123–128.
[4] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7399400/
[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[6] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Comput.*, vol. 12, no. 4, pp. 40–49, Oct./Dec. 2013.
[7] S. Wang, L. Le, N. Zahariev, and K. K. Leung, "Centralized rate control mechanism for cellular-based vehicular networks," in *Proc. IEEE Globecom Workshops(GCWorks)*, Dec. 2013, pp. 4914–4920.
[8] M. Patel *et al.*, "Mobile-edge computing—Introductory technical white paper," ETSI, Sophia-Antipolis, France, Tech. Rep., 2014, no. 1. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf
[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.
[10] S. Wang *et al.*, "Mobile micro-cloud: Application classification, mapping, and deployment," in *Proc. Annu. Fall Meeting ITA (AMITA)*, 2013, pp. 1–8.
[11] V. Bahl, "Cloud 2020: The emergence of micro datacenter for mobile computing," Microsoft, Redmond, WA, USA, Tech. Rep., 2015. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/11/Micro-Data-Centers-mDCs-for-Mobile-Computing-1.pdf
[12] K. Ha *et al.*, "Adaptive VM handoff across cloudlets," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-113, 2015.
[13] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.
[14] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperation in 5G wireless networks: An SDN approach," *IEEE Netw.*, vol. 31, no. 5, pp. 14–20, Sep. 2017.
[15] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun. (2015). "Fog computing: Focusing on mobile users at the edge." [Online]. Available: https://arxiv.org/abs/1502.01815
[16] M. Chiang and W. Shi, "NSF workshop report on grand challenges in edge computing," Nat. Sci. Found., Alexandria, VA, USA, Tech. Rep., 2016. [Online]. Available: http://iot.eng.wayne.edu/edge/NSF%20Edge%20Workshop%20Report.pdf
[17] D. Chen *et al.*, "S2M: A lightweight acoustic fingerprints-based wireless device authentication protocol," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 88–100, Feb. 2017.
[18] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 603–608.
[19] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
[20] Y. Kawamoto, H. Nishiyama, N. Kato, N. Yoshimura, and S. Yamamoto, "Internet of Things (IoT): Present state and future prospects," *IEICE Trans. Inf. Syst.*, vol. 9, no. 10, pp. 2568–2575, 2014.
[21] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1457–1477, 3rd Quart., 2017.
[22] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.
[23] B. Panja, D. Morrison, P. Meharia, B. Bharat, and A. Prakash, "Group security of V2V using cloud computing processing and 4G wireless services," *Int. J. Next-Generat. Comput.*, vol. 5, no. 3, pp. 26–31, 2014.
[24] J. Chen *et al.*, "Service-oriented dynamic connection management for software-defined Internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2826–2837, Oct. 2017.
[25] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.
[26] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Migrating running applications across mobile edge clouds: Poster," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2016, pp. 435–436.
[27] S. Fu *et al.*, "Interference cooperation via distributed game in 5G networks," *IEEE Internet Things J.*, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8014424/
[28] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.
[29] R. Urgaonkar *et al.*, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, Sep. 2015.
[30] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Commun. Mag.*, vol. 50, no. 9, pp. 34–40, Sep. 2012.
[31] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implementation (NSDI)*, 2005, pp. 273–286.
[32] J. Shetty, M. R. Anala, and G. Shobha, "A survey on techniques of secure live migration of virtual machine," *Int. J. Comput. Appl.*, vol. 39, no. 12, pp. 34–39, 2012.
[33] H. Alshahrani, A. Alshehri, R. Alharthi, A. Alzahrani, D. Debnath, and H. Fu, "Live migration of virtual machine in cloud: Survey of issues and solutions," in *Proc. Int. Conf. Secur. Manage. (SAM)*, 2016, pp. 280–285.
[34] Q. Huang, F. Gao, R. Wang, and Z. Qi, "Power consumption of virtual machine live migration in clouds," in *Proc. 3rd Int. Conf. Commun. Mobile Comput. (CMC)*, Apr. 2011, pp. 122–125.
[35] N. Ekiz, T. Salih, and S. Küçüköner, and K. Fidanboylu. (2005). *An Overview of Handoff Techniques in Cellular Networks*. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.192.9836
[36] S. Pack, X. Shen, J. W. Mark, and J. Pan, "Mobility management in mobile hotspots with heterogeneous multihop wireless links," *IEEE Commun. Mag.*, vol. 45, no. 9, pp. 106–112, Sep. 2007.
[37] J. Han and B. Wu, "Handover in the 3GPP long term evolution (LTE) systems," in *Proc. Global Mobile Congr. (GMC)*, Oct. 2010, pp. 1–6.
[38] W. Bao and B. Liang, "Stochastic geometric analysis of handoffs in user-centric cooperative wireless networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[39] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 1291–1296.
[40] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.
[41] A. Gani, G. M. Nayeem, M. Shiraz, M. Sookhak, M. Whaiduzzaman, and S. Khan, "A review on interworking and mobility techniques for seamless connectivity in mobile cloud computing," *J. Netw. Comput. Appl.*, vol. 43, pp. 84–102, Aug. 2014.
[42] N. Zhang, N. Cheng, A. T. Gamage, K. Zhang, J. W. Mark, and X. Shen, "Cloud assisted HetNets toward 5G wireless networks," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 59–65, Jun. 2015.
[43] H. Zhang, X. Chu, W. Guo, and S. Wang, "Coexistence of Wi-Fi and heterogeneous small cell networks sharing unlicensed spectrum," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 158–164, Mar. 2015.
[44] J. Li *et al.*, "On social-aware content caching for D2D-enabled cellular networks with matching theory," *IEEE Internet Things J.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8025784/
[45] 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., Services (MobiSys)*, 2013, pp. 153–166.
[46] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep./Oct. 2013.
[47] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1350–1354.
[48] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Oct. 2014, pp. 835–840.

[49] K.-H. Chiang and N. Shenoy, "A 2-D random-walk mobility model for location-management studies in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 53, no. 2, pp. 413–424, Mar. 2004.

[50] S. Wang, R. Urgaonkar, K. Chan, T. He, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 5504–5510.

[51] K. LaCurts, J. C. Mogul, H. Balakrishnan, and Y. Turner, "Cicada: Introducing predictive guarantees for cloud networks," in *Proc. Usenix Conf. Hot Topics Cloud Comput. (HotCloud)*, 2014, pp. 1–14.

[52] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1994.

[53] R. Langar, N. Bouabdallah, and R. Boutaba, "A comprehensive analysis of mobility management in MPLS-based wireless access networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 918–931, Aug. 2008.

[54] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Comput. Netw.*, vol. 57, no. 9, pp. 2093–2115, 2013.

[55] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 1082–1090.

[56] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[57] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[58] K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "Live data center migration across WANs: A robust cooperative context aware approach," in *Proc. SIGCOMM Workshop Internet Netw. Manage. (INM)*, 2007, pp. 262–267.

[59] K. Ha and M. Satyanarayanan, "Openstack++ for cloudlet deployment," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-123, 2015.

[60] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[61] T. K. Refaat, B. Kantarci, and H. T. Mouftah, "Dynamic virtual machine migration in a vehicular cloud," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2014, pp. 1–6.

[62] D. Montero and R. Serral-Gracià, "Offloading personal security applications to the network edge: A mobile user case scenario," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Sep. 2016, pp. 96–101.

[63] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," in *Proc. 10th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS)*, 2016, pp. 258–269.

[64] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito, "Exploring container virtualization in IoT clouds," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, May 2016, pp. 1–6.

[65] P. Angin and B. K. Bhargava, "An agent-based optimization framework for mobile-cloud computing," *JoWUA*, vol. 4, no. 2, pp. 1–17, 2013.

[66] P. Angin, B. Bhargava, and Z. Jin, "A self-cloning agents based model for high-performance mobile-cloud computing," in *Proc. IEEE 8th Int. Conf. Cloud Comput. (CLOUD)*, Jun./Jul. 2015, pp. 301–308.

[67] P. Angin, B. Bhargava, and R. Ranchal, "Tamper-resistant autonomous agents-based mobile-cloud computing," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 843–847.

[68] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[69] S. Alami-Kamouri, G. Orhanou, and S. Elhajji, "Overview of mobile agents and security," in *Proc. IEEE Int. Conf. Eng. MIS (ICEMIS)*, Sep. 2016, pp. 1–5.

[70] F. Luo, Y. Chen, Z. Xu, G. Liang, Y. Zheng, and J. Qiu, "Multiagent-based cooperative control framework for microgrids' energy imbalance," *IEEE Trans. Ind. Inform.*, vol. 13, no. 3, pp. 1046–1056, Jul. 2017.

[71] X. Zhu, C. Chen, L. T. Yang, and Y. Xiang, "ANGEL: Agent-based scheduling for real-time tasks in virtualized clouds," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3389–3403, Dec. 2015.

[72] M. E.-K. Fareh, O. Kazar, M. Femmam, and S. Bourekkache, "An agent-based approach for resource allocation in the cloud computing environment," in *Proc. 9th Int. Conf. Telecommun. Syst. Services Appl. (TSSA)*, Nov. 2015, pp. 1–5.

[73] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, F. Xia, and S. A. Madani, "Virtual machine migration in cloud data centers: A review, taxonomy, and open research issues," *J. Supercomput.*, vol. 71, no. 7, pp. 2473–2515, 2015.

[74] Z. Liu, W. Qu, W. Liu, and K. Li, "Xen live migration with slowdown scheduling algorithm," in *Proc. Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT)*, Dec. 2010, pp. 215–221.

[75] 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./Dec. 2009.

[76] H. Yao, C. Bai, D. Zeng, Q. Liang, and Y. Fan, "Migrate or not? Exploring virtual machine migration in roadside cloudlet-based vehicular cloud," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 18, pp. 5780–5792, 2015.

[77] P. Lu, A. Barbalace, and B. Ravindran, "HSG-LM: Hybrid-copy speculative guest os live migration without hypervisor," in *Proc. 6th Int. Syst. Storage Conf. (SYSTOR)*, 2013, pp. 2–9.

[78] S. Simanta, G. A. Lewis, E. Morris, K. Ha, and M. Satyanarayanan, "A reference architecture for mobile code offload in hostile environments," in *Proc. Int. Conf. Mobile Comput., Appl., Services (MobiCASE)*, Aug. 2012, pp. 274–293.

[79] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[80] B. Paten *et al.*, "The NIH BD2K center for big data in translational genomics," *J. Amer. Med. Informat. Assoc.*, vol. 22, no. 6, pp. 1143–1147, 2015.

[81] A. Mirkin, A. Kuznetsov, and K. Kolyshkin, "Containers checkpointing and live migration," in *Proc. Linux Symp.*, vol. 2. 2008, pp. 85–90.

[82] K. M. Sim, "Agent-based cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, Oct. 2012.

[83] H. Liang, B. J. Choi, W. Zhuang, X. Shen, A. S. A. Awad, and A. Abdr, "Multiagent coordination in microgrids via wireless networks," *IEEE Wireless Commun.*, vol. 19, no. 3, pp. 14–22, Jun. 2012.

[84] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multiagent Systems With JADE*, vol. 7. Hoboken, NJ, USA: Wiley, 2007.

[85] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.

[86] N. Ahmed and B. K. Bhargava, "Towards dynamic QoS monitoring in service oriented archtectures," in *Proc. Int. Conf. Cloud Comput. Services Sci. (CLOSER)*, 2015, pp. 163–171.

[87] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[88] C. A. Kamienski, F. F. Borelli, G. O. Biondi, I. Pinheiro, I. D. Zyrianoff, and M. Jentsch, "Context design and tracking for IoT-based energy management in smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 687–695, Apr. 2017.

[89] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[90] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct./Dec. 2015.

[91] M. Nir, A. Matrawy, and M. St-Hilaire, "Economic and energy considerations for resource augmentation in mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 99–113, Jan./Mar. 2015.

[92] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[93] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[94] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 10037–10047, Dec. 2016.

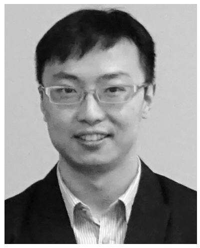
[95] A. Stanciu, "Blockchain based distributed control system for edge computing," in *Proc. 21st Int. Conf. Control Syst. Comput. Sci. (CSCS)*, May 2017, pp. 667–671.

[96] Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* [Online]. Available: https://bitcoin.org/en/bitcoin-paper

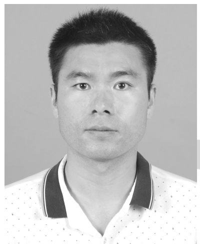
[97] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13rd USENIX Symp. Netw. Syst. Design Implementation(NSDI)*, 2016, pp. 45–59.

SHANGGUANG WANG (SM'11) received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT) in 2011. He is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, BUPT. He has co-authored over 100 papers. His research interests include edge computing, service computing, and cloud computing. He is the Vice Chair of the IEEE Computer Society Technical Committee on Services Computing and the President of the Service Society Young Scientist Forum in China. He has played a key role at many international conferences, such as a General Chair of CollaborateCom 2016, a General Chair of ICCSA 2016, and a TPC Co-Chair of IEEE EDGE 2018.

JINLIANG XU received the bachelor's degree in electronic information science and technology from the Beijing University of Posts and Telecommunications in 2014, where he is currently pursuing the Ph.D. degree in computer science with the State Key Laboratory of Networking and Switching Technology. His research interests include mobile cloud computing, service computing, information retrieval, and crowdsourcing.



NING ZHANG received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, under the supervision of Prof. J. W. Mark, in 2015. After that, he was a Post-Doctoral Research Fellow at the University of Waterloo and the University of Toronto, under the supervision of Prof. S. (Xuemin) Shen and Prof. B. Liang, respectively.



YUJIONG LIU received the M.E. degree in computer science and technology from Chongqing University in 2010. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include mobile cloud computing, edge computing, and Internet of Things.