Path Selection for Seamless Service Migration in Vehicular Edge Computing

Jinliang Xu, Xiao Ma*, *Member, IEEE*, Ao Zhou, *Member, IEEE*, Qiang Duan, *Member, IEEE*, Shangguang Wang, *Senior Member, IEEE*

***Abstract*—Mobile edge computing provisions computing and storage resources by deploying edge servers (ESs) at the edge of the network to support ultra-low delay and high bandwidth services. To ensure QoS of latency-sensitive services in vehicular networks, service migration is required to migrate data of the ongoing services to the closest ES seamlessly when users move across different ESs. To achieve seamless service migration, path selection is proposed to obtain one or more paths (consisting of several switches and ESs) to transfer service data. We focus on the following problems about path selection: Where to implement path selection? How to coordinate interests of mobile users (i.e., vehicles) and network providers since they have conflicting interests during path selection? How to ensure seamless service migration during migration of vehicles? To address the above problems, this paper investigates path selection for seamless service migration. We propose a path selection algorithm to jointly optimize both interests of network plane (i.e., cost for network providers) and service plane (i.e., QoE of users). We first formulate it as a multi-objective optimization problem and further prove theoretically that the proposed algorithm can give a *weakly Pareto optimal solution*. Moreover, to improve the scalability of the proposed algorithm, a distance-based filter strategy is designed to eliminate undesired switches in advance. We conduct experiments on two synthesized datasets and the results validate the effectiveness of the proposed algorithm.**

***Index Terms*—mobile edge computing, vehicular networks, service migration, path selection, Pareto optimal.**

I. Introduction

Mobile edge computing (MEC) supports ultra-low delay and high bandwidth services for vehicular networks by deploying edge servers (ESs), which is also named as micro-clouds [1] or MEC hosts [2], at the edge of network. While vehicles [3], [4] and smart phones [5] always move around [6], [7], [8], limited coverage of each ES can result in dramatic drop of QoS or even service interruption [1]. To this end, stateful service data needs to be transferred from the source ES to the target one (the closest one), i.e., service migration in vehicular edge computing [2]. If the incurred service interruption lasts no more than a specific time threshold so that users cannot directly observe its occurrence, it is considered as *seamless service migration* [9]. An application scenario of service migration of a moving vehicle in vehicular edge computing is exemplified in Fig. 1. When the vehicle moves from the left to right along the road, the network path changes from BS1-ES1 to ES1-S1-S2-S5-S7-BS6, ES1-S1-S2-S4-S6-S7-BS6, etc. The network distance becomes much longer than the beginning, resulting in network QoS degradation for users [10]. Service migration is promising to address this problem in vehicular edge computing. By migrating the ongoing service data from ES1 to the closest ES5, the network path reduces to BS6-ES5. During service migration, at least one feasible transferring path should be determined before transferring service data (i.e., path selection), e.g., ES1-S1-S2-S5-S7-ES5, ES1-S1-S2-S4-S6-S7-ES5 or ES1-S1-S3-S5-S7-ES5. Path selection attempts to transfer data efficiently from the source ES to the target one, by properly selecting one or more feasible transferring paths.

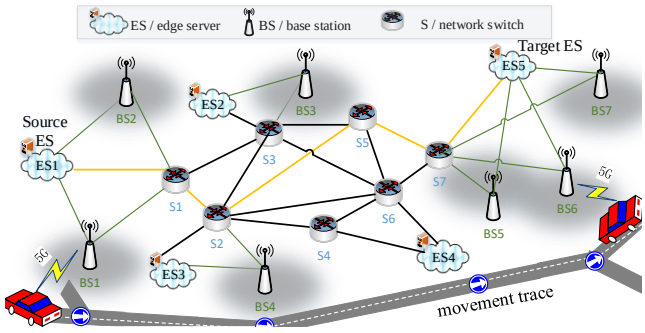


Fig. 1. Service migration and data transferring path as a vehicle moves along the road in vehicular edge computing. A network link may exists between two network switches or ESs, and the ones with orange color make up a feasible data transferring path in a service migration from source ES1 to target ES5.

We propose to implement path selection logic in the traffic steering module of the 5G MEC architecture. As shown in Fig. 2, 5G MEC provides a unified control for networks and services through software defined network (SDN) and network function virtualization (NFV) technology [11], [12]. Traffic steering serves as a centralized entity to control all of the ESs and network switches in the vehicular edge computing system, and it is proposed to route traffic of application instances in 5G MEC. SDN controller directs the switches to deliver network services wherever they are needed, regardless of the specific connections between network nodes. It is quite different from traditional network architecture, in which individual network switches make traffic decisions based on their configured routing tables, i.e., in a decentralized decision-making way. An ES in 5G MEC includes a user plane function (UPF) [2], [13]. UPF is a fundamental component of 5G. And it represents the

Jinliang Xu, Xiao Ma, Ao Zhou, and Shangguang Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. E-mail: jlxu@bupt.edu.cn; maxiao18@bupt.edu.cn; aozhou@bupt.edu.cn; sgwang@bupt.edu.cn. Xiao Ma is the corresponding author.

Qiang Duan is with the Information Science and Technology Department, The Pennsylvania State University, Abington, PA 19001. E-mail: qduan@psu.edu.

xxxx-xxxx/0x/$xx.00 2020 IEEE Published by the xxxx xxxx

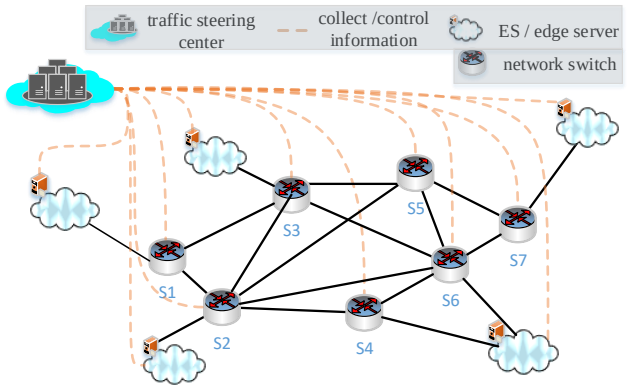


Fig. 2. Traffic steering of 5G MEC connects to multiple ESs and network switches, collect information from them, and route traffic according to the results of the proposed path selection algorithm to transfer data for service migration.

evolutionary trend that separates the control plane and user plane. So it works well with application interfaces that are implemented in SDN manner. As a result, by connecting to the UPF of each ES, traffic steering can collect resource demand (including network, computing and storage resource, etc.) of mobile users. Moreover, it can also obtain the load information of every network switch in the core network, and then route traffic among ESs and network switches according to established rules. In short, traffic steering of 5G MEC can realize the function of path selection during service migration to select the best data transferring paths, and then transfer service data according to the found transferring paths. However, there are problems remain to be resolved: 1) Path selection in service migration concerns two parameters, namely QoS for mobile users (e.g., delay of each network link), and network cost from network providers (price of each network link). Mobile users and network providers are two rational stakeholders, and they have conflicting interests. So it is needed to research how to make a balance between them; 2) While transferring service data of one service, if more than one transferring paths work simultaneously, more data will be transferred to the target ES within given time period. So it is important to know how many transferring paths should be selected, and how to find them; 3) If data transferring time in a service migration is too large, mobile users will observe QoS degradation or even service interruption [9]. The ideal result is seamless service migration, and its transferring time is less than a threshold time. So it is important to research how to find a set of transferring paths that need time less than threshold time to transfer all of the service data.

Our contributions are summarized as follows. 1) We propose to use traffic steering in 5G MEC as the platform to run the proposed path selection algorithm in a SDN manner; 2) We build a two-layer system model (namely service layer and network layer), and propose to build them into a single mathematic model jointly like a multi-objective optimization. Service plane focuses on minimizing the time cost for transferring data from source to target ES [9]. The second layer is network plane. For a network service provider, the target of path selection is how to reduce the network cost in the case of service migration [14]; 3) We combine parameters of the two-layer system model by weighted sum, and prove that it can give a weakly Pareto optimal solution of the original problem; 4) We choose a set of paths instead of just one to improve efficiency of data transferring. The proposed algorithm aims to choose the best set of available transferring paths that can minimize the total transferring time with limited bandwidth of each network connection; 5) Time threshold for seamless service migration is introduced in the proposed path selection algorithm. It makes proper path selection for vehicular edge computing as mobile users are more sensitive to transferring time.

The remainder of this paper is organized as follows. In Section II we introduce in detail the proposed path selection algorithm on delay and price in service migration of mobile edge computing. Section III describes our experimental setup and results on two synthesized datasets, and validates the effectiveness of the proposed algorithm. A review of related work is given in Section IV. Finally, the conclusion and future work are laid out in Section V.

II. Path Selection Algorithm in Traffic Steering

We detail the proposed path selection algorithm, the analysis of the Pareto optimal solution and the scalability problem.

A. Preliminary

We introduce two kinds of parameters, one for service plane and another for network plane on a 5G MEC network. As shown in Fig. 1, there are several nodes (ESs or network switches) on a transferring path from source ES to target ES. A connection between two neighboring nodes on a path is called a *network link*, e.g., $l_{1,1}$ between ES1 and network switch S1, $l_{1,3}$ between S1 and S3. So transferring path ES1-S1-S3-S5-S7-ES5 can be represented by several sequentially connected network links as $(l_{1,1}, l_{1,3}, l_{3,5}, l_{5,7}, l_{7,5})$. For two arbitrary nodes i and j with direct connection with each other, their network link $l_{i,j}$ has three attributes, namely bandwidth $b_{i,j}$, delay $t_{i,j}$ (i.e., the time for one bit to transmit from one end of $l_{i,j}$ to the other.) and price $p_{i,j}$ (i.e., the money to be paid for one unit of data volume transferred on $l_{i,j}$.). In this work, we consider delay t as a service plane parameter because users are interested in network delay as one important QoS, and price p as a network plane parameter because network providers are interested in reducing network cost when maintaining a network.

As price and delay are items on two different planes and not easy to optimize simultaneously, by involving them together we construct a new parameter named network cost. Both of delay and price of the network links should be taken into account when we build the model for joint optimization of network plane and service plane. First we normalize the delay and price attributes using *feature scaling*. And it brings all values into the range $[0, 1]$. The normalizing process is formulated as follows:

$$t'_{i,j} = \frac{t_{i,j} - t_{min}}{t_{max} - t_{min}}, \qquad p'_{i,j} = \frac{p_{i,j} - p_{min}}{p_{max} - p_{min}}, \tag{1}$$

where symbols t_{max} and t_{min} are the maximum and minimum values of the delay of the network links respectively, then p_{max}

and p_{min} are the maximum and minimum values of the price of the network links respectively, at last $t'_{i,j}$ and $p'_{i,j}$ are the delay and price values after being normalized. Then we build the concept of *network cost* $c_{i,j} = w_t t'_{i,j} + w_p p'_{i,j}$ for each network link. Network delay, price and the newly constructed cost can be written down in matrix form, i.e., delay matrix T, price matrix P, and cost matrix C. T is as follows:

$$T = \begin{pmatrix} 0 & t'_{1,2} & \cdots & t'_{1,N} \\ t'_{2,1} & 0 & \cdots & t'_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ t'_{N,1} & t'_{N,2} & \cdots & 0 \end{pmatrix}, \tag{2}$$

where N denotes the total number of nodes except for mobile devices (including ESs and network switches). P is constructed in the same way as T. Note that delay matrix T and price matrix P are not symmetric (i.e., not necessarily $t'_{i,j} = t'_{j,i}$ and $p'_{i,j} = p'_{j,i}$ for arbitrary i, j pair) due to the fact that the network links are not necessarily symmetric [15]. Moreover, the matrices T and P may be highly sparse as most computation entities are not directly connected. The building of cost matrix C is by:

$$C = w_t T + w_p P, \tag{3}$$

where $w_t + w_p = 1, w_t, w_p \in [0, 1]$, C is the cost matrix containing $N \times N$ elements, and w_t and w_p are the weights of delay and price respectively.

B. Path Selection Algorithm

Transferring time is another parameter of service plane, and it introduces the time threshold for seamless service migration. The former is relative to the data volume to be transferred and the bandwidth of every network link on a transferring path, and the latter is relative to the bad impact on the QoS if the data transferring time is too large in a service migration case. In some mobile applications like digital reading, if the time is less than $0.1s$, the mobile user is imperceptible; while if the respond time is in range $(0.1, 1)s$, the mobile user will find that it is difficult to maintain user's train of thought; while if the respond time is larger than 10s, the mobile user will find that it is difficult to feel appropriately connected [9], [16]. So we use symbol $\mathring{T}$ to denote the time threshold for seamless service migration. If the data transferring time is less than $\mathring{T}$, the possible QoS degradation by service migration will not perceived by mobile users. Note that the transferring time is different from the network delay time, which means the time for a bit to travel from the source to the target network node.

The path selection algorithm is shown in Alg. 1. The main idea is to find a new transferring path in each cycle and then update the remaining bandwidth resource of each link on this path, until these paths together can transfer all data volume in time $\mathring{T}$; if no new path can be found in a cycle, we relax $\mathring{T}$ to make sure that in the new time period all data volume can be transferred. Dijkstra algorithm is adopted in each cycle to find a transferring path between the source ES and the target server, and the path has the minimum sum cost of its all links (Lines 2-3) [17]. Dijkstra is feasible because the cost of a network link (Eq. 3) is relative to its delay and price, and both of them are additive. Lines 5-7 describe the relaxation of time

Algorithm 1: Path selection algorithm on delay and price in service migration.

Input: O_s, source ES; O_d, target ES; Q, the data volume needed to be transferred from O_s to O_d; $\mathring{T}$, the time threshold for seamless service migration; the set of network links $\{l_{i,j} = \langle b_{i,j}, c_{i,j} \rangle\}$, of which the subscript i, j represents an arbitrary pair of ESs or network switches;

Output: The best set of transferring paths S; the estimated transferring time $\mathring{t}$.

1 Initialize the set of transferring paths $S \leftarrow \{\}$;
2 consider the cost matrix defined network of network switches between O_s and O_d as a weighted graph;
3 apply Dijkstra algorithm to the weighted graph to find the shortest path s from O_s to O_d;
4 **if** *s not exists* **then**
5 compute $B \leftarrow \sum_{s \in S} b_s^{min}$, where b_s^{min} means the *bandwidth of a transferring path* s in set S;
6 compute the practical transferring time $\mathring{t} \leftarrow Q/B$, where Q is the data volume to be transferred;
7 return S and $\mathring{t}$;
8 update $S \leftarrow S \cup \{s\}$;
9 get b_s^{min}, i.e., *bandwidth* of s as Definition 1 ;
10 **if** $b_s^{min} \mathring{T} > Q$ **then**
11 return S and $\mathring{T}$;
12 **else**
13 subtract b_s^{min} from the bandwidth of each link on s, e.g., for $l_{i,j} \in s$, we have $b_{i,j} \leftarrow b_{i,j} - b_s^{min}$;
14 delete all of links with zero bandwidth value;
15 **if** $\mid S \mid = 1$ **then**
16 update $Q' \leftarrow Q - b_s^{min} \times \mathring{T}$;
17 **else**
18 update $Q' \leftarrow Q' - b_s^{min} \times \mathring{T}$;
19 goto line 2;

threshold $\mathring{T}$. As the initial time threshold is the maximum time limit for a seamless service migration, its relaxation means that seamless service migration cannot be realized. The task of relaxation operation is to find the minimum time period that can transfer all data volume. $\mathring{t}$ in Line 6 is the minimum practical transferring time. This is ensured by Proposition 1. Lines 13-18 show how to update the bandwidth resource of each network link on the new found path in each cycle. Line 10 means the condition that the found paths together can transfer all data volume in time $\mathring{T}$. Line 4 means the condition that no new path can be found in a cycle.

Definition 1 (Bandwidth of Transferring Path)**.** *A transferring path is made up of several sequentially connected network links. Every network link has its bandwidth value, and the bandwidths of different networks may be different. We define the bandwidth of the transferring path as the minimum bandwidth value of all network links on the transferring path.*

Proposition 1. $\mathring{t} = Q / \sum_{s \in S} b_s$ *is the minimal time cost needed to transfer a piece of Q-sized data by making use*

of a set of independent transferring paths denoted by symbol S, where b_s means bandwidth of transferring path s.

Proof 1: Given an arbitrary transferring path, let it be $s_1 \in S$, the transferring time t_1 on it is less than $\mathring{t}$, i.e., $t_1 < \mathring{t}$. According to pigeonhole principle [18], there is at least one transferring path s with transfer time of $t_s \geq \frac{Q-b_1 t_1}{\sum_{s\in S} b_s - b_1} > \frac{Q-b_1 \mathring{T}}{\sum_{s\in S} b_s - b_1} = \mathring{t}$. That is to say, there will never be a case where the transferring time of all paths are less than $\mathring{t}$. So $\mathring{t}$ is the minimal time cost needed. ∎

The best case of the complexity of Alg. 1 is $O(N^2)$, and the worst case is $O(N^4)$. As we know, the complexity of Dijkstra algorithm is $O(N^2)$ if the number of nodes in the input graph is N. The largest number of links among N node is $\frac{1}{2}N(N-1)$. In the best case, the first transferring path meets the needs, which results in complexity of $O(N^2)$. While in worst case, in each cyle just one link is deleted, until all links are gone. So the complexity will be $O(N^4)$. If we denote the numer of found transferring path as K, the complexity of Alg. 1 can be written as $O(KN^2)$.

In this work, only one path selection is to be made during one time-slot. It cannot coordinate the multiple requirements very well. If more than one user migrates the edge servers simultaneously, the output of the algorithm will be not necessarily ideal. The reason is that as the network resources (e.g., network bandwidth) are always limited, if two users of two service migration requirements are close enough, they may act on each other. However, we think this case will not always happen. The reason is as follows: as we have mentioned, the time slot is small enough. So the moving distance of a user during this time slot will be very small. The input network resources needed for one service migration requirement by the path selection algorithm are always in a small area. In a small area, the possibility of more than one requirements is very unlikely. Moreover, if we consider coordination of multiple requirements, the needed computation time will grow rapidly. If the time needed exceeds the size of the time slot, the output of the algorithm will be unreliable. While in vehicle edge computing, the time slot must be small enough to ensure the real-time performance.

C. Pareto Optimal Solution Analysis

Although the path selection algorithm does not take delay and price directly as inputs, it balances both demands from network plane and service plane because of Eq. 3. In this section we discuss its relation from optimizing delay and price separately.

First we give definitions of two kinds of problems, and then we discuss their relations. In Alg. 1, we apply Dijkstra algorithm to the weighted graph to find the shortest path s from source ES to target ES. The problem of Dijkstra algorithm can be formulated by the following (***Problem I***):

$$s = \arg\min_{s\in S} O_C(s), \tag{4}$$

where symbol $O_C(s)$ denotes a function that applies Dijkstra algorithm on the weighted graph C to find a shortest path s. Eq. 3 shows that cost matrix C is a weighted sum of delay matrix T and price matrix P with weights w_t and w_p respectively. As the Dijkstra algorithm is totally a linear function, *Problem I* can be transformed into a weighted sum of two sub-problems by the following:

$$s = \arg\min_{s\in S} w_t O_T(s) + w_p O_P(s). \tag{5}$$

We define a multi-objective optimization problem as follows (***Problem II***):

$$s = \arg\min_{s\in S} \left[O_T(s), O_P(s)\right], \tag{6}$$

which means finding a transferring path $s \in S$ that is able to minimize $O_T(\cdot)$ and $O_P(\cdot)$ at the same time. Note that Eq. 5 and Eq. 4 are equal, and they are two kinds of expressions of Problem I. While Eq. 6, i.e., Problem II, is quite different from Problem I. So Eq. 5 can not be transformed into Eq. 6 directly. A multi-objective problem is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Problem II means finding a transferring path $s \in S$ that is able to minimize $O_T(\cdot)$ and $O_P(\cdot)$ at the same time. So it is a multi-objective problem.

In this work function O_C is not written down with a function expression. However, Dijkstra algorithm is determined. And the proposed Alg. 1 is determined. So as long as the involved network parameters are fixed, including the network topology, the bandwidths, price and cost of every involved network link, function O_C will be determined. The algorithm runs in a time slot that is much less than the time threshold, and in this time slot the data is transferred according to the output paths set. As the time slot is quite short, the involved network parameters can be considered as constant during the data transferring. The time slot can be short enough as a data transferring task can be completed in more than one time slot.

In the following, we give the definitions of Pareto optimal solution and weakly Pareto optimal solution. Next we study the relations between the solutions of *Problem I* and *Problem II*. They are demonstrated by two propositions (Propositions 2 and 3) and the proofs following each of them.

Definition 2 (Pareto Optimal Solution)**.** *$s^* \in S$ is said to be a Pareto optimal solution of Problem II, if and only if there does not exist another $s \in S$, such that $O_T(s) \leq O_T(s^*)$ and $O_P(s) \leq O_P(s^*)$.*

Definition 3 (Weakly Pareto Optimal Solution)**.** *$s^* \in S$ is said to be a Pareto optimal solution of Problem II, if and only if there does not exist another $s \in S$, such that $O_T(s) < O_T(s^*)$ and $O_P(s) < O_P(s^*)$.*

Proposition 2. *The solution of Problem I is a weakly Pareto optimal solution of Problem II.*

Proof 2: Let $\hat{s} \in S$ be a solution of Problem I. Let us suppose that it is not a weakly Pareto optimal solution of Problem II. In this case, there exists a solution $s \in S$ such that $O_T(s) < O_T(\hat{s})$ and $O_P(s) < O_P(\hat{s})$. According to the assumptions set to the weights $w_t \geq 0$ and $w_p \geq 0$ for at least one is larger than zero. Thus $w_t O_T(s) + w_p O_P(s) < w_t O_T(\hat{s}) + w_p O_P(\hat{s})$. This is a contradiction to the assumption that $\hat{s}$ is a solution of Problem I. Or the solution $\hat{s}$ must be a weakly Pareto optimal solution of Problem II. ∎

Proposition 3. *The solution of Problem I is a Pareto optimal solution of Problem II if the weights of the two sub-problems in Problem II are both positive, i.e.,* $w_t > 0$ *and* $w_p > 0$.

Proof 3: Let $\hat{s} \in S$ be a solution of Problem I with positive weights. Let us suppose that is not Pareto optimal. This means that there exists a solution $s \in S$ such that $O_T(s) \leq O_T(\hat{s})$, $O_P(s) \leq O_P(\hat{s})$ and $O_T(s) < O_T(\hat{s})$, $O_P(s) < O_P(\hat{s})$ holds for at least one. Since $w_t > 0$ and $w_p > 0$, we have $w_t O_T(s) + w_p O_P(s) < w_t O_T(\hat{s}) + w_p O_P(\hat{s})$. While this result contradicts the assumption that $\hat{s}$ is a solution of Problem I. That means solution $\hat{s}$ must be Pareto optimal. ■

From the above analysis, we find that the weighted sum in Eq. 3 is reasonable for traffic steering of service migration in vehicular edge computing. Concretely, the method cannot find a transferring path that can at the same time minimize delay and price, though, it can ensure a weakly Pareto optimal solution. Further, if both of the weights are positive values, it can find a Pareto optimal solution.

D. Scalability Problem using an Elliptic Region Based Method

The scalability problem arises when the coverage of vehicular network expands. Suppose the number of network nodes per unit area in urban area is constant, and the length of the city is d, then the number N is proportional to d^2. The time complexity of Alg. 1 can also be written down as $O(Kd^4)$, and $O(d^8)$ at worst. The high complexity will result in several terrible consequences when mobile users moves in a city: 1) executing the path selection algorithm may be prohibitively expensive (e.g., time cost and energy consumption); 2) it costs much more time to collect information; 3) as the cost time become larger, the accuracy of the path selection algorithm will be heavily discounted because the time slot for Alg. 1 must be small enough.

To relieve this problem, an elliptic region method is adopted to filter out the inappropriate network nodes before executing the path selection algorithm. In mathematics, an ellipse is a plane curve surrounding two focal points, such that for all points on the curve, the sum of the two distances to the focal points is a constant; for any point outside the curve, the sum distance is larger than the constant; and for an any point inside the curve, the sum distance is less than the constant [19]. Specifically, treat the source and target ESs in a service migration as two elliptical focuses; then use them to draw an ellipse; only nodes inside the ellipse are selected as the input of the path selection algorithm. It can help to filter out most of the network nodes, and the algorithm will scale well when deployment density of the nodes is high or the distance between source to target ESs is large.

The rationality of the elliptic region method for scalability can be justified with the general characteristics of network node distribution. A longer transferring path always means more complex network condition, more network switches and longer network delay, which is detrimental to the data transferring in service migration. Only one network node serving as the intermediate node between the source to target ESs is the most likely case in data transferring of service migration.

Fig. 3. An example of elliptic region based method on base stations from Shanghai Telecom Industry. A bubble in the picture means a base station deployment at its place. This is an enlarged image for a certain position of the map at the bottom right corner. And it is a heat map that shows the Shanghai overview of distribution density of base stations.

If the sum of the network distance from the intermediate node to the source and target ESs is fixed, or smaller than the fixed value for better QoS, all of the possible intermediate nodes are inside an elliptic region.

The elliptic region method brings about the need of balancing the size of the elliptic region, which can be represented by the ratio of the sum distance and the distance between two focal points. An experimental show of the covering area of the elliptic region based method on real data of the base stations from Shanghai Telecom Industry is in Fig. 3. If the size is too large, the elliptic region will cover too many nodes; otherwise no sufficient transferring paths will be found, and we have to turn to the relaxation of time threshold. Neither can ensure a seamless service migration. The size of the ellipse should be determined from experience or multiple experiments. In replicate experiments conducted by us on the Telecom data, if the size of ellipse is large enough to cover 50 network nodes, it can balance the two aforementioned conflicting goals very well.

III. Experimental Evaluation

From last section, we know that the proposed path selection method can theoretically give a weakly Pareto optimal solution about the network plane and service plane. While the theoretical analysis alone is not enough. So experimental tests on datasets are needed to validate the proposed method.

A. Dataset and Platform

As no specific off-the-shelf dataset exists, we choose to generate synthesized data from the following two real datasets. Note that in the proposed method, the distance between two nodes is network distance, i.e., hops between them. For the limit of the datasets, we still use geographical distance to represent the network distance. This simplification makes sense for two reasons: 1) if the geographical distance is large, a single transferring path will be involved in more data transferring requests from mobile users. This will result in small bandwidth, long netowrk delay, etc.; 2) as the geographical distance becomes larger, there will be more network nodes on a transferring path. This means larger number of hops [6]. If the network nodes are the same, larger hops always mean long delay [20].

We get two public datasets to generate two synthesized datasets, and the latter is used for experimental tests. Two public datasets are: 1) Data Traffic Records of Shanghai[1]. This data contains a lot of mobile base stations in a large city with their locations (i.e., latitude and longitude) and workloads (i.e., the number of connected mobile users in a time slice); and 2) Vehicles' GPS Traces of Shanghai. The data of vehicles' GPS Traces [3] is also recorded in Shanghai. One piece of trace data contains a sequence of locations and the correspondent time stamps of a taxi in its moving. An the two synthesized datasets are generated as follows:

1) DATA #1 is generated from *Data Traffic Records of Shanghai*. As an ES or network node can be attached to a cellular base station in vehicular edge computing, a base station in the datasets can be considered as an ES or network node. The bandwidth value of a network link follows the uniform distribution between $[0.5, 1]$, and the same goes for the price. As the delay of a link is relative to its length, we set the delay of a link as the distance between the two network nodes. We select two base stations with the maximum network distance as the source and target ESs.
2) DATA #2 is generated from both of the above two real datasets. As the two datasets come from the same city, we can combine them. We know a taxi's distance from any ES at any time when it moves. One piece of trace data shows that the corresponding taxi moves from one location to another. At the start point, we can find the nearest ES. We consider it as the source ES. At the end point, we can find the nearest ES as the target ES. We selected 50 traces as DATA #1.

The majority of the code is implemented in Python 2.7. The experiments are all performed on a server with a 16-core 2.6 GHz Intel Xeon processor with 32 GB of RAM.

B. Baseline Algorithms and Metrics

We list the proposed algorithm and three baselines, and their relations as follows:

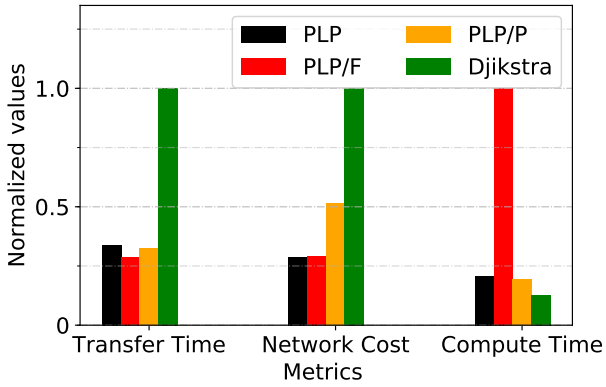
- PLP. The proposed algorithm. It is about **p**ath selection on **l**atency and **p**rice in service migration.
- PLP/F. A simplified version of PLP. It is without elliptic region based method for filtering out network nodes.
- PLP/P. Another version of PLP without taking price into account, i.e., delay matrix acting as cost matrix.
- Dijkstra.It is used inside PLP to find the shortest path.

Obviously the time complexity of PLP/F and PLP/P is the same to PLP, i.e., $O(KN^2)$, while the time complexity of Dijkstra is $O(N^2)$.

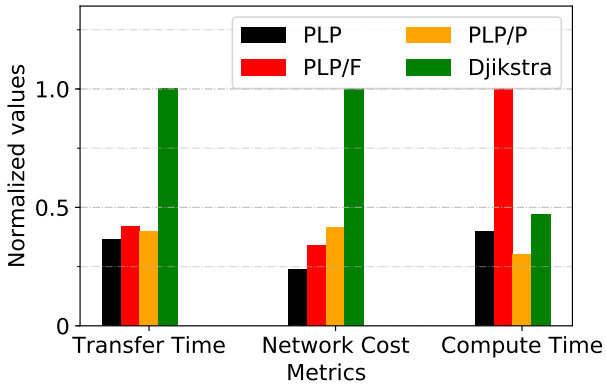
We employ the following three evaluation metrics to evaluate the performance of the proposed path selection algorithm:

- *Transfer Time*. It is needed to transfer all of the data from the source to target ES. It is relevant to the data volume and network bandwidth.
- *Network Cost*. The network cost is paid by network provider to transfer all of the data from the source to target ES.

[1]http://sguangwang.com/TelecomDataset.html



(a) DATA #1



(b) DATA #2

Fig. 4. Experimental results on two datasets.

- *Compute Time*. The time needed to run the algorithms. It is relevant to the time cost and energy consumption.

A lower value of the above three evaluation metrics implies a good performance. Note that for each data, we evaluate a set of source-target ES pairs, and compute the mean value as the final result in the three metrics.

C. Experimental Results

1) DATA #1: The experimental results are as shown in Fig. 4(a). Note that the values for each metric are normalized to a range of $[0, 1.0]$ by being divided by the maximum value. The maximum values for transfer time, network cost, and compute time are $2.29s$, 42.33, and $0.73ms$, respectively. As we can see, Dijkstra has the best performance in computing time, but the worst with both of transferring time and cost at the same time. Dijkstra produces only a single transferring path with the shortest length. It costs the least computing time because Dijkstra terminates when it finds the shortest path. However, the shortest path does not necessarily have the lowest price. What is more, the bandwidth of a single path is always enough, because adding more path will generate a large bandwidth sum and take less transferring path. Compared to the proposed PLP, PLP/F has significant increase in computing time. That is because that PLP/F must choose transferring path from a much large candidate set than PLP. While at the same time, the performance in transferring time and cost of PLP/F is just a little better than the proposed PLP. So we argue that the proposed PLP performs better in general than PLP/F. PLP/P

TABLE I
AFFECTS OF ELLIPTIC REGION SIZE

Metrics	Region Size = 4				Region Size = 8				Region Size = 12			
	PLP	PLP/F	PLP/P	Dijkstra	PLP	PLP/F	PLP/P	Dijkstra	PLP	PLP/F	PLP/P	Dijkstra
Transfer Time (s)	1.18	1.35	1.28	3.23	1.17	1.35	1.28	3.13	1.17	1.35	1.28	3.13
Network Cost (1)	14.78	21.23	25.9	62.59	15.1	21.23	46.1	60.7	14.7	21.23	54.39	60.7
Compute Time (ms)	0.33	0.83	0.25	0.39	6.77	0.83	6.8	4.77	44.3	0.83	52.9	39

does a little better in transferring time and computing time, but performs much worse in cost than the proposed PLP. The reason is that PLP/F does not take price into account when it gives the paths. From Fig. 4(a), we can conclude that the proposed PLP performs much better than the three baseline algorithms.

2) DATA #2: The experimental results of four methods on three metrics are as shown in Fig. 4(b). The maximum values for transfer time, network cost, and compute time are $3.23s$, 62.59, and $0.83ms$, respectively. It shows a similar result in comparison to Fig. 4(a). Dijkstra has the best performance in computing time, but the worst with both of transferring time and cost at the same time. This is because Dijkstra produces only a single transferring path with the shortest length. Compared to the proposed PLP, PLP/F performs much worse in computing time and cost. That is because PLP/F must choose transferring path from a much large candidate set than PLP. While at the same time, the performance in transferring time of PLP/F is in the same scale with the proposed PLP. PLP/P does a little better in computing time, but perform worse in cost and transferring time than the proposed PLP. So we conclude that the proposed PLP performs better as it takes both of delay and cost of network connections into account.

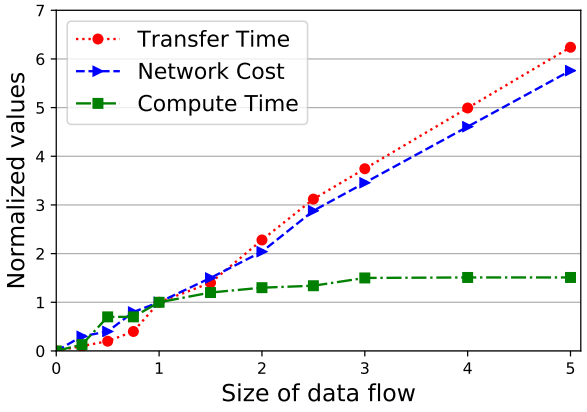


Fig. 5. Affects of Dataflow size.

3) Affects of Dataflow Size: We also conduct simulation experiments to show the affects of dataflow size on the above-mentioned three metrics, and the results are shown in Fig. 5. The data size in the first experiment is considered as one unit here. The metric values are normalized by dividing the values of PLP with one unit data size on DATA #1, namely 0.77s on tranfer time, 12.11 on network cost, and 0.15ms on compute time. When the size of data to be transferred is zero, the values of three metrics return to zero. When the data size is less than one unit, the growing speeds of transfer time and network cost become larger. The reason is that the new selected transferring path always has larger network cost and network latency. When the data size exceeds 3 units, the compute time does not grow any more, and the transfer time and network cost are in proportion to the data size. This phenomenon means the occurrence of relaxation of time threshold in Alg. 1. If this happen, no new transferring path would be found any more. The results are obtained on DATA #1, and using DATA #2 we can get similar results.

4) Affects of Elliptic Region Size: The effect of elliptic region size is as shown in Table I. We set parameter *Region Size* as 4, 8 and 12 respectively, and conduct the experiments on DATA #2 to obtain the experimental results. Note that PLP/P does not filter out ESs, so changing of elliptic region size has no influence on it. Dijkstra still performs badly on cost and transferring time. The proposed PLP generally performs better. As elliptic region size increases, the performance comparison between PLP and PLP/F will approach one another. The reason is that when the elliptic region increase to an extent, the number of ESs in the elliptic region is enough to find the best transferring scheme.

The experimental results show that the proposed path selection algorithm cannot minimize the data transferring time, network cost, and computing time. However, it does pretty well in these three metrics. As a result, we can conclude that both of experimental results and the theoretical analysis in the previous section validate the effectiveness of our work.

IV. RELATED WORK

The standard for MEC system in 5G is still under development[2], and there are not yet very solid works on traffic steering or service migration. However, traffic steering is an import aspect in the future 5G [2] to gain high efficiency in utilizing network. Traffic steering can serve as infrastructures of service migration in a 5G MEC system. It is quite different from traditional data transferring tasks between network switches due to its application scenes and evaluation metrics [21]. Its differences from the traditional cellular handoff are discussed in a survey paper of service migration [21]. Cellular handoff is used to avoid the interruption of telecommunication services when mobile users move across cellulars. In comparison to the general SDN applications [22], Traffic steering and

[2]https://www.3gpp.org/release-17

service migration mainly focus on how to cooperate multiple geographically-distributed ESs in 5G MEC.

Traffic steering can be realized using SDN technology, and path selection in service migration can be realized using traffic steering. Traffic steering in MEC refers to the capability of the MEC system to route traffic to the targeted applications in a distributed cloud. Whilst in a generic MEC architecture as defined in [23], traffic steering is controlled by the MEC platform through configuring the data plane. In a 5G integrated deployment the role of the data plane is delegated to UPF of 5G. A UPF plays the central role in routing the traffic to desired applications and network functions [2]. 5G will provide a unified control based on SDN and NFV to run the proposed traffic steering algorithm [24]. For example, the control structure of path selection in traffic steering can run on OpenStack platform, as OpenStack with SDN module can control large pools of compute, storage, and networking resources that are distributed at many geographically dispersed ESs [25], [26].

Path selection is indispensable to traffic steering. The path selection problem stems from the basic tradeoff between the cost of service migration (transmission cost and migration cost) and the improvement of users' expectation on QoS that can be achieved after migration [27], [1], [28]. If the target ES is determined, another problem is to minimize the total time used for migrating the service data. Andrew et al. [29] proposed to use a three-layered framework to avoid transferring unnecessary data if the needed data has a copy at the target ES. Ha et al. [27] proposed to make use of various compression algorithm to reduce the size of transferred data. These works focus on how to minimize the data volume to be transferred in the service migration process. But data compression ratio has its inherent limit, and will take additional time. So the use of these works is limited.

It is important to optimize jointly the network plane (cost of network provider) and service plane (network delay and data transferring time), which is a point of our work. No matter to what extent the transferred data is minimized, it must be sent out into the mobile network [7], [27], [1]. If we choose the best set of transferring paths, the total transferring time will be minimized considerably. Although Ha et al. [27] found that bandwidth of network connection has a great influence on the time cost of service migration, they have not further utilized the mobile network to improve the service migration. In addition, from the aspects of network operator, 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 [14]. Work in [30] tries to jointly optimize the content placement and content delivery in the vehicular edge computing network using a Markov decision process method. While the path selection problem in our work can be considered as static in the limited time slot, so we do not take into account other external factors. While it is promising to solve the service migration based problems in an integrated model, such as when to start a service migration, where is the target ES, and how to transfer data.

Our work is also different from multi-path TCP (MPTCP) [31]. MPTCP is a general multi-path solution for efficiency and robustness, instead of being customized for 5G MEC. The time threshold for seamless service migration in 5G MEC is an important concept in the proposed path selection algorithm, which can take into account both of the network plane and service plane.

V. Conclusion

Traffic steering is an important aspect of a 5G MEC system, and it is a proper tool to run path selection algorithm to realize seamless service migration in vehicular edge computing. This paper recognizes the importance of path selection in traffic steering, and then proposes a path selection algorithm to jointly optimize in both of network plane and service plane. It is oriented to data transferring at edge network with limited bandwidth and price of each network connection. We conduct theoretical analysis and experiments against baseline algorithms, and the results show that the proposed algorithm can help to alleviate QoS degradation in service migration.

In this work, the involved entities like edge servers and network switches are supposed to be owned by one company or an enterprise alliance. It is reasonable to conduct information gathering, global optimization, and centralized control based on SDN and NFV technology. However, in 5G, more wireless access technologies can be utilized for collaboration of edge servers, network switches, and mobile users, such as Super Wi-Fi, fixed wireless, etc. This can lead to demand of reorganization and the flexibility of the network nodes and data migrating services, especially when more third-party edge servers or network switches join the vehicular edge computing system [32]. It is important to focus on trust problem and collaboration way of the many entities in a geographically-distributed vehicular edge computing system. Thus, we will study how to solve these inherent problems using decentralized technology like blockchain smart contract.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61922017, 61902036), Funds for Creative Research Groups of China (61921003) and China Postdoctoral Science Foundation 2019M650589.

References

[1] 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 Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017.

[2] S. Kekki, R. Arora, L. M. Contreras, Y. Fang, W. Featherstone, and D. Frydman, "MEC in 5G networks," *ETSI white paper*, vol. 1, no. 28, pp. 1–28, 2018.

[3] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 919–928, ACM, 2010.

[4] K. Xiong, S. Leng, J. Hu, X. Chen, and K. Yang, "Smart network slicing for vehicular fog-rans," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3075–3085, 2019.

[5] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, *et al.*, "Mobile-edge computing introductory technical white paper," *Mobile-edge Computing (MEC) Industry Initiative White Paper*, 2014.

[6] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proceedings of 33rd IEEE Military Communications Conference (MILCOM)*, pp. 835–840, IEEE, 2014.
[7] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 1, pp. 1–14, 2016.
[8] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proceedings of the 37th IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–9, IEEE, 2018.
[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 2–11, 2009.
[10] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, 2017.
[11] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
[12] A. Aissioui, A. Ksentini, and A. Gueroui, "An efficient elastic distributed SDN controller for follow-me cloud," in *Proceedings of the 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 876–881, IEEE, 2015.
[13] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.
[14] F. Zhao and X. Zeng, "Optimization of user and operator cost for large-scale transit network," *Journal of Transportation Engineering*, vol. 133, no. 4, pp. 240–251, 2007.
[15] Z. Tang, Z. Wang, P. Li, S. Guo, X. Liao, and H. Jin, "An application layer protocol for energy-efficient bandwidth aggregation with guaranteed quality-of-experience," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1538–1546, 2015.
[16] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5853–5863, 2019.
[17] M. Sniedovich, "Dijkstra's algorithm revisited: the dynamic programming connexion," *Control and Cybernetics*, vol. 35, no. 3, p. 599, 2006.
[18] M. Ajtai, "The complexity of the pigeonhole principle," in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 346–355, IEEE, 1988.
[19] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Scale and orientation adaptive mean shift tracking," *IET Computer Vision*, vol. 6, no. 1, pp. 52–61, 2012.
[20] M. Haenggi and D. Puccinelli, "Routing in ad hoc networks: a case for long hops," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 93–101, 2005.
[21] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.
[22] F. Foresta, W. Cerroni, L. Foschini, G. Davoli, C. Contoli, A. Corradi, and F. Callegati, "Improving openstack networking: Advantages and performance of native sdn integration," in *Proceedings of the 52nd IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.
[23] M. ETSI, "Mobile edge computing (MEC); framework and reference architecture," *ETSI, DGS MEC*, vol. 3, 2016.
[24] 5G-PPP, "5g vision brochure v1," *5G-PPP white paper*, vol. 1, pp. 1–11, 2016.
[25] D. Haja, M. Szabo, M. Szalay, A. Nagy, A. Kern, L. Toka, and B. Sonkoly, "How to orchestrate a distributed openstack," in *Proceedings of the 37th IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 293–298, IEEE, 2018.
[26] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
[27] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, and M. Satyanarayanan, "Adaptive VM handoff across cloudlets," *Technical Report CMU-CS-15-113*, pp. 1–27, 2015.
[28] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Computer Networks*, vol. 108, pp. 357–370, 2016.
[29] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Migrating running applications across mobile edge clouds: poster," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 435–436, ACM, 2016.
[30] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 247–257, 2020.
[31] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp.," in *NSDI*, vol. 11, pp. 8–8, 2011.
[32] J. Xu, S. Wang, B. Bhargava, and F. Yang, "A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3538–3547, 2019.

Jinliang Xu received the bachelor degree in electronic information science and technology from Beijing University of Posts and Telecommunications in 2014. Currently, he is a Ph.D. candidate in computer science at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include Mobile Cloud Computing, Blockchain, AI, and Crowdsourcing.

Xiao Ma received her Ph.D. degree in Department of Computer Science and Technology from Tsinghua University and B.S. degree in Telecommunication Engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2018 and 2013. She is currently a postdoctoral fellow at the State Key Laboratory of Networking and Switching Technology, BUPT. From October 2016 to April 2017, she visited the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include mobile cloud computing and mobile edge computing.

Ao Zhou is an associate professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She received her Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2015. Her research interests include cloud computing, edge computing, and service reliability.

Qiang Duan received the BS degree in electrical and computer engineering and the MS degree in telecommunications and electronic systems. He received the PhD degree in electrical engineering from the University of Mississippi. He is an associate professor at the Pennsylvania State University Abington College. His currently active research areas include future internet architecture, network virtualization, network-as-a-service, software defined network, and cloud computing. He has published over 70 papers and authored five book chapters. He is on the editorial boards for more than 10 international research journals and has served on the technical program committees for numerous international conferences.

Shangguang Wang received his PhD degree at Beijing University of Posts and Telecommunications in 2011. He is Professor and Vice-Director at the State Key Laboratory of Networking and Switching Technology (BUPT). He has published more than 150 papers recent years, and played a key role at many international conferences, such as general chair and PC chair. His research interests include service computing, cloud computing, and mobile edge computing. He is a senior member of the IEEE in 2011, and Editor-in-Chief of the International Journal of Web Science.