AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections.

AQ1: Please confirm or add details for any funding or financial support for the research of this article.
AQ2: Please cite "Table I" inside the text.
AQ3: Please provide descriptions of the labeled subparts for the captions of Figs. 7 and 8.
AQ4: Please confirm the volume number for Reference [3].
AQ5: Please provide the organization name for the B.S.Eng. degree attained by the author S. Mahmood.

5G-Enabled MEC: A Distributed Traffic Steering for Seamless Service Migration of Internet of Vehicles

Muhammad Rizwan Anwar, Shangguang Wang, *Senior Member, IEEE*, Muhammad Faisal Akram, Salman Raza, and Shahid Mahmood

***Abstract*—Multiaccess edge computing (MEC) is considered as a backbone for the 5G network. The successive MEC network combines the networking and computation at the edge of the network to achieve the Quality of Services (QoS) with ultralow latency. The devices with mobility feature, whether hand-held devices or vehicles move from one edge server (ES) location to another ES, creates a nonoptimal environment in the long run. To maintain QoS and avoid service disruptions, existing network topologies do not fulfill the requirement. Hence, a unique traffic steering with dynamic path selection is required for live service migration of time-sensitive applications. In this article, we are the first to introduce a distributed traffic steering through the differentiation of two different types of network elements (i.e., ESs and routers), in a large MEC system. Using this concept, we, for the first time, resolve the scalability problem of a large MEC network into a partitioned MEC network. The proposed framework bounds the path-finding procedure with a filter strategy based on the network distance to eliminate the excess of non-related network elements. With a decentralized framework for MEC, we propose matrix-based dynamic shortest path selection and matrix-based dynamic multipath searching algorithms for dynamic path selection under the proposed autonomous network boundary discovery and must connect node block benchmarks. Our proposed dynamic traffic steering system works under two most important metric measurements (time delay and available bandwidth). Experimental results validate the effectiveness of dynamic and adaptive path searching in a partitioned controlled MEC network that significantly outperforms the centralized approaches with 35%–70% efficiency in QoS.**

***Index Terms*—5G, collaborative mobile-edge computing, live service migration, path routing, path selection, traffic steering.**

I. Introduction

THE MULTIACCESS edge computing (MEC) optimizes the performance for ultralow latency and high throughput with efficient bandwidth availability using the edge server (ES) as the backbone of the 5G network. ES is also considered as the mobile-cloud [1] or MEC server host [2], [3]. The MEC system is a key requirement for the continuity of application during user mobility whether traditional hand-held devices or vehicles movement [4]. Especially, it has become very important in the case of a fast-moving autonomous vehicle's application. For instance, an application serving the user may need to be transferred to another ES with the user movement at a new location. Consequently, for state-full applications, the service state data can be transferred from the origin ES to the destination ES [5]. A consistent service migration framework takes the responsibility to migrate the ongoing services, seamlessly, from the source to target ES near the current user's location. The researchers prove that live service migration resolves this problem by moving real-time service data from the origin ES to the destination ES [5], [6]. Therefore, the threshold time for both computation of origin destination (OD) ES path and transferring the service data should not to be exceeded than the required minimum time slot [7]. Hence, a seamless service migration remains challenging, when the MEC network expands to a large network.

The rapid advancement in MEC and 5G technology efficiently resolve this problem, by deploying user plan function in MEC network servers (ES). This is an essential module of a 3GPP 5G core infrastructure and represents the data plane evolution of a control and user plane separation scheme [8]. According to the 5G future plan [2], the traditional network measurement and monitoring protocols, such as OSPF and IS-IS [9], cannot cover all the requirement of dynamic traffic monitoring in MEC. So, a distributed traffic steering system becomes a key requirement for Quality-of-Services (QoS) network traffic management in the 5G MEC network. The main target of traffic steering is to maintain both data transferring and service connectivity in minimum time slot with low error rate.

In this article, for the first time, we devise a dynamic path selection philosophy in static and decentralized traffic steering structure of the 5G-enabled MEC framework. We contest this challenge utilizing divide-and-conquer strategy by both unified and distributed control for MEC network functions [2] and service availability through the software-defined network (SDN) and network function virtualization (NFV) [10], [11]. Although, there are proposed studies on traffic steering in MEC, they are fundamentally incompatible with the time-complexity problem of this article. First, the researcher focused only unified control and use centralized approach

Manuscript received November 16, 2020; revised March 15, 2021; accepted May 24, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1805502; in part by NSFC under Grant 61922017 and Grant 61921003; and in part by the Open Research Fund of Key Laboratory of Space Utilization, Chinese Academy of Sciences under Grant LSUKFJJ-2019-03. *(Corresponding author: Shangguang Wang.)*

Muhammad Rizwan Anwar, Shangguang Wang, Muhammad Faisal Akram, and Salman Raza are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: m.r.anwar@bupt.edu.cn; sgwang@bupt.edu.cn; m_faisal15@hotmail.com; salmanraza@bupt.edu.cn).

Shahid Mahmood is with the Electronics Engineering Department, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: shahid13m809@gmail.com).

Digital Object Identifier 10.1109/JIOT.2021.3084912

that always consider a full graph of the MEC network. Second, previous works are usually based on machine learning approaches that do not fulfill the time-complexity problem due to high computational requirements [6], [7], [12], [13]. While, traffic steering can serve as a distributed entity with SDN functionality at every ES, instead of the centralized control server [7].

Throughout our research, we address and summarize the challenges of traffic steering in the MEC system.

1) Path finding in live service migration that entails QoS for mobile users with the selection of minimum delay and higher bandwidth requirement.
2) The higher bandwidth and lowest delay have a conflicting sympathy. So, it needs to research on how to balance both of these metrics.
3) Throughout the service migration process, if multiple paths can select at once, more data can be transferred to the destination server within the minimum time slot. So, it becomes more important to concentrate how many paths can be searched and how to find multiple paths dynamically in a minimum time period.
4) During transferring whole service data, if migration time is too large, the users will face QoS interruption [5]. There is essential how to manage a decentralized traffic steering procedure that can provide less than the threshold time with fully optimal path searching for whole service data to transfer.

In this research, our contributions are concise as follows.

1) We propose two Branch-and-Bound algorithms.
   a) An autonomous network boundary discovery (ANBD) that generates a partition-based immutable controlled matrix on the current server, which is represented by a Bounded (B) benchmark. Using the ANBD algorithm at a MEC server, the current ES can see its adjacent ESs only.
   b) To avoid the nonrelated servers and routers during path searching, we introduce a must connect node block (MCNB) algorithm that creates immutable lists of must connected nodes (MCNs) and is known by the Mapped (M) benchmark, where each node list is represented by its adjacent MEC server.

   The most important point is that both of these modules are used only once at deployment time of MEC servers or after restarting the server.
2) We, for the first time, introduce the importance of communication and processing delay separately with a significance of the available bandwidth to compute as a single mathematical model jointly, like a multiobjective optimization.
3) We give analyses to combine these parameters in the path selection system by weighted sum and it is proved that it can provide a full optimal solution of the original problem.
4) We also introduce two dynamic path searching algorithms based on immutable resultant features of ANBD and MCNB. First, a matrix-based dynamic shortest path selection (MDSPS) algorithm, which finds a single shortest best path, using a related set in MCNB, which has the lowest delay cost and optimized bandwidth. Second is the matrix-based dynamic multipath searching (MDMPS) algorithm between OD pair of ESs.
5) The time threshold for smooth live service data migration is introduced, to make fast and proper path finding for autonomous vehicle communication to ES that is considered more sensitive in transferring time.

The remainder of this research work is as follows. In Section II, we formally introduce the 5G-enabled MEC framework with an optimization model and its analysis with the Pareto-optimal solution. We define a large network problem, in Section III, with two branch-and-bound algorithms for the solution of the problem. We describe our two proposed dynamic path searching algorithms based on the decentralized solution, in Section IV. Section V presents our evaluation of the solution and results with and without solving the large-scale network problem. A summarized review is given with related research in Section VI. At the end, a brief closure with future work is described in Section VII.

II. Preliminary

In this section, we first describe the system model, notions, and notations. We then introduce the joint optimization model for path selection and parameter analysis precisely.

A. MEC Network Model

In this section, we present a 5G MEC network system model, where one MEC server is associated directly with one or more BSs. The number of BSs is related to the MEC deployment [14], which is beyond the scope of our research. Fig. 1 demonstrates an example of the MEC network with several nodes as MEC servers and routers (E_1, E_2, and E_3 and R_1, R_2, and R_3, respectively). The scenario in Fig. 1 demonstrates a service migration example from origin ES_1 to a corresponding destination ES_2 and then next to ES_5. The network distance and congestion becomes much larger with longer distance from ES_1 to ES_5. Even, security-based autonomous vehicular real-time application cannot continue from ES_1 to ES_2. This scenario results in network QoS degradation [15].

This whole synchronous MEC network can be represented as the simple undirected graph $G = (N, L)$, where $N = \{n_1 \dots n_r\}$ is the set of nodes of the network and $L = \{l_1 \dots l_p\}$ is the set of links between the connected nodes. Each node from G corresponds to the nodes (MEC servers and routers both) of the MEC network, and the links $L \in G$ reflect the fact that pairs of nodes are sequentially adjacent. Note that for the cardinalities $|N| = r$ and $|L| = p$, the MEC network G has r nodes and p links, respectively, in a data path. We only consider three basic metrics $(c_{i,j}, w_{i,j}, a_{i,j})$ of link $l_{i,j} \in L$ in network G. For a notational convenience, $c_{i,j}$ is used for the total capacity of link communication, $w_{i,j}$ denotes the network latency or link weight and $a_{i,j}$ represents the available bandwidth, corresponding to the link $l_{i,j} \in L$. Specifically, if the nodes $n_i \in N$ and $n_j \in N$ adjoin to the consecutively adjacent connected pair by the link $l_{i,j} \in L$, then: $w_{i,j} = defT_{\mathrm{delay}}(n_i, n_j \in N)$. The main difference is that the

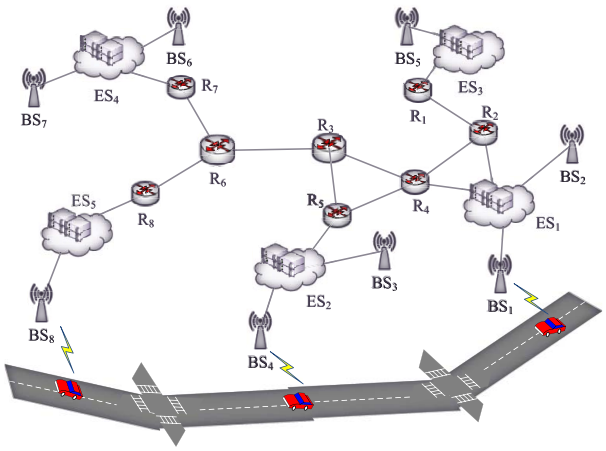

Fig. 1. Service connectivity as a vehicle going right to left along the road in vehicular edge computing. The distance becomes longer with movement going out from initially connected areas.

TABLE I
SUMMARY OF NOTATIONS

Sets	
G	MEC network graph
N	Routers and Servers as node ($N \subset G$)
L	links between two nodes ($n_i \rightarrow n_j$)
H	Possible sub-networks ($H \subset G$)
M	Edge Servers ($M \subset N$)
R	MEC Network Routers ($R \subset N$)
Network Graph Parameters	
m_o	Origin ES ($m_o \in M$)
m_d	Destination ES ($m_d \in M$)
m_a	A set of adjacent ESs ($m_a \in M$)
$\|x\|$	Cardinality of a set x
p_i	Specific path between OD ESs ($p_{O\rightarrow D}$)
P	set of paths $P_{O\rightarrow D} = \{p_1, p_2 \ldots p_k\}$
$[C_{i,j}]_{n\times n;\ n=\|N\|}$	Whole MEC network matrix
$[C^H_{i,j}]_{n\times n;\ n=\|H\|}$	Sub-network graph matrix
$[C^R_{i,j}]_{n\times n;\ n=\|MCN\|}$	Possible reduced matrix through MCN
Demand Parameters	
$D_p^{o\rightarrow d}$	Possible delay of path
$B_p^{o\rightarrow d}$	Possible bandwidth of path
$D^l_{i,j}$	Cumulative delay of link ($w_{i,j} + a^{-1}_{i,j}$)
K	Data size of service to be transferred
T_p	Practical transferring time of path
$\overline{T}$	Threshold time of service data to transfer
Network Parameter	
$c_{i,j}$	Link capacity between node $n_i \rightarrow n_j$
$w_{i,j}$	Normalize delay of a link $n_i \rightarrow n_j$
$a_{i,j}$	Available bandwidth of a link $n_i \rightarrow n_j$
η_i	Processing state of i^{th} node ($\eta_i \in n_i \in N$)
$a^{-1}_{i,j}$	Multiplicative inverse of bandwidth

link-cost of a link $l_{i,j} \in L$ received by one node $n_i \in N$ could be different from another node $n_j \in N$ since this information is disseminated in an asynchronous manner [9].

With the use of feature scaling, we normalize the delay $w_{i,j}$, for the convenience of subsequent processing. It passes all delay values $w'_{i,j}(t)$ in the range [0; 1]. The normalization function is formulated as follows:

$$w_{i,j} = \frac{w'_{i,j}(t) - w'_{i,j}(t)\,\min}{w'_{i,j}(t)\,\max - w'_{i,j}(t)\,\min} \tag{1}$$

where symbols $w'_{i,j}(t)\,\min$ and $w'_{i,j}(t)\,\max$ are defined as the network delay or data transferring time. In such a MEC model, $p_{o\rightarrow d}$ is the set of all available links between OD pair of nodes. Similarly, the minimum distance from $p_{o\rightarrow d}$ is time-dependent, denoted by $D_p^{o\rightarrow d}(t)$, and can be expressed as following in our model:

$$D_p^{o\rightarrow d}(t) = \min \sum_{\forall l_{i,j}\in p} w_{i,j} + \eta_i \tag{2}$$

where we consider, separately, the link propagation delay $w_{i,j}(t)$ and all types of node (router and MEC servers) processing delay $\eta_i(t)$ at time t [16].

On the other hand, the entire path $p_{o\rightarrow d}$ contains $l_{i,j}$, which is the ith link in the MEC network graph G. Likewise, $c_{i,j}$ is the capacity of $l_{i,j}$, $b_{i,j}$ is its current load, and $a_{i,j}$ is the related available capacity of bandwidth, which becomes simple as

$$a_{i,j} = c_{i,j} - b_{i,j}. \tag{3}$$

Furthermore, it is assumed here that the capacity $c_{i,j} \in l_{i,j}$ is known with practical assumption by the Link State Advertisement [9] system. If any additional policy changes the bandwidth on the ith link, the network operator should be informed about it. The application should be able to compute the current bandwidth load $b_{i,j}$ of directly connected links at all nodes ($\forall l \in N$). This can be applicable by an approach similar as previously presented in [17]. We can periodically calculate the current load $b_{i,j}(t)$ at time t as

$$b_{i,j}(t) = \frac{\hat{x}_i(t) - \hat{x}_i(t-\tau)}{\tau} \tag{4}$$

where $\hat{x}_i(t)$ is the counter value and τ is the time interval.

Although, we consider the bandwidth availability at $a_{i,j}$, as unused capacity even the link $l_{i,j} \in L$ is idle or transmitting the packets at the maximum speed [17]. The available bandwidth a_i must be looked as the average unused capacity and has a great influence in service migration [2]. Using the bandwidth load $b_{i,j}$, the available bandwidth $a_{i,j}$ can be modeled as the following equation [18]:

$$a_{i,j}(t,\tau) = \frac{1}{\tau}\int_t^{\tau+t} (c_{i,j} - b_{i,j}(t))\mathrm{dt} \tag{5}$$

where $a_{i,j}(x)$ is the immediately available bandwidth at a given time x at the link $l_{i,j}$. Much research [17]–[19] has been proposed about delay and available bandwidth value collection and can be studied in detail.

Note that we calculate the delay cost of a path as the total cost of all links in a path set ($\forall l_{i,j} \in p_i$). Whereas, the concept of bandwidth of transferring path is different and is defined in Definition 1. In the case, we required to find a path $p_{o\rightarrow d}$ in network G having the highest bandwidth of path $B_p^{o\rightarrow d}$. All the procedures can be calculated with the following equation:

$$B_p^{o\rightarrow d}(t) = \max_{p_{o\rightarrow d}} \min_{l_{i,j}\in p} a_{i,j}. \tag{6}$$

Definition 1 (Bandwidth of linked Path): A network path is a set of several connected links as $p = \{l_{1,2}, l_{2,3}, l_{3,5} \ldots\}$, for data transferring. Every link $l_{i,j} \in p$ has its assigned bandwidth value $c_{i,j} \in l_{i,j}$, and in different network links, it may vary. We

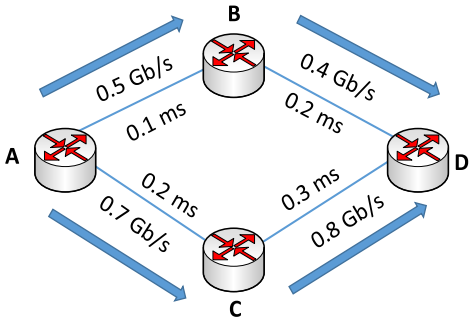

Fig. 2. Two different paths between the OD pair and shortest-path selection based on link delay and bandwidth inverse.

define the available bandwidth of a linked path as the minimum available bandwidth $a_{i,j}^{\min} \in p$ value from all links on the path.

B. Joint Optimization Model

The delay cost functionality performs as the lesser link delay cost that specifies the higher bandwidth availability at a specific link. In contrast, the higher the cost signifies, the lesser the bandwidth availability [18]. The problem arises, when a link $l_{i,j} \in p$ is selected with low latency but the bandwidth capacity is lower than the required data to be transferred and it will affect the QoS. As we know, the efficient bandwidth availability is more important for seamless service migration, we introduce here an objective function to achieve better results in path selection

$$D_p^{o\to d}(t) = \min \sum_{\forall l_{i,j}\in p} w_{i,j} + a_{i,j}^{-1} + \eta_i \tag{7}$$

where $a_{i,j}^{-1}$ is the multiplicative inverse function of the available bandwidth value that can be considered as a bandwidth cost of a link [20]. For instance, a specified link $l_{i,j} \in p$ provides 10 MB and the other link $l_{i,k} \in p$ has 100-MB available bandwidth then the multiplicative inverse of both is 1/10 = 0.1 and 1/100 = 0.001, respectively. With the consideration of Fig. 2, there are two baths $p_1 = A \to B \to D$ and $p_2 = A \to C \to D$ from nodes A–D having different capacity at each link. Thus, the available bandwidth of p_1 signifies as 0.4 Gb/s; whereas, p_2 is as 0.7 Gb/s (we use Mb/s values as Gb/s to maintain bandwidth values between [0; 1]). The path p_1 is selected as the shortest path if we consider only link delay cost with a total cost of 0.3 ms than path p_2 cost, which is 0.5 ms. Whereas, the available bandwidth of path p_1 is 0.4 Gb/s, which is lower than the p_2 bandwidth of 0.7 Gb/s. This situation means the degradation of QoS in optimal path selection. The multiplicative inverse of the available bandwidth of each link fixes this problem with an additive value of both link delay and bandwidth inverse as follows:

$$D_{i,j}^{l}(t) = \sum w_{i,j} + a_{i,j}^{-1}. \tag{8}$$

Additive values of each link through the above equation specify the path p_2 as the shortest path with a cost of 3.9 and the available bandwidth of 0.7 Gb/s, while p_1 cost is calculated as 4.8. This approach achieves higher quality results in a single shortest path. So, we take into account the cumulative technique for the link cost in (7).

In the sense of the above discussion, the whole MEC network G states can be described as an $N \times N$ matrix, C is denoted as $[C_{i,j}]_{n\times n}$; $n = |N|$ is a symmetric matrix containing the graph connectivity. All elements of cost matrix $C_{i,j}$, in general, can be defined as

$$C_{i,j} = \begin{cases} \eta_{i,j}, & \text{if } i = j \\ \{w_{i,j}, a_{i,j}\}, & \text{if } i \neq j \\ \text{Null}, & \text{Otherwise.} \end{cases} \tag{9}$$

Here, we, for the first time, identify the importance of processing delay and status of every type of nodes (servers and routers), for QoS dynamic path selection and collaborative MEC network. The states of all nodes ($\forall \eta_i \in N$) represents the status of memory and CPU availability with the help of both queuing and processing delays [16] and significantly can help in traffic load management, which will be discussed in our next research work. As in (9), all diagonal elements ($[C_{i,j}]_{n\times n}, i = j$) signify the additive processing states as $\eta_{i,j}$. Whereas, all nondiagonal values ($[C_{i,j}]_{n\times n}, i \neq j$) represent the available bandwidth $a_{i,j}$ and the delay cost $w_{i,j}$, of the ith link, between nodes $n_i \in N$ and $n_j \in N$. Here, we set $([C_{i,j}]_{n\times n}, i \neq j) = Null$ that signifies no direct connection between the nodes $n_i \in N$ and $n_j \in N$.

C. Parameter Comparison and Analysis

In this section, we talk about the important relation between link delay $w_{i,j} \in p$ and available bandwidth $a_{i,j} \in p$ and prove their joint optimization with the Pareto-optimal relation. First, we define two types of problems and then we elaborate relations of both. In dynamic path searching, a weighted graph is used to search the shortest best path $p_{o\to d}$ from the origin ES to destination ES. The problem related to the path searching algorithm can be expressed by the following (Problem I):

$$p = \arg\min_{p\in P} O_C(p) \tag{10}$$

where symbol $O_C(p)$ denotes a function that utilizes the MDSPS algorithm on the cost matrix C with cumulative values of delay cost ($w_{i,j} \in p$) and bandwidth cost ($a_{i,j}^{-1} \in p$) to find the best shortest path p. As the MDSPS algorithm works as a linear function, Problem I can transform into a weighted sum of two subproblems as the following equation:

$$p = \arg\min_{p\in P} C_w O_w(p) + C_a O_a(p) \tag{11}$$

where $C_w O_w(p)$ denotes the function for the cost of link delay ($w_{i,j} \in l_{i,j} \in p$) and $C_a O_a(p)$ as a function for bandwidth cost ($a_{i,j}^{-1} \in l_{i,j} \in p$) . As well as, it can be defined as a multiobjective optimization problem as follows (Problem II):

$$p = \arg\min_{p\in P} \left[O_w(p), O_a(p)\right]. \tag{12}$$

It means searching a path $p \in P$, which is able to minimize $O_w(x)$ and $O_a(x)$ at once. Note that the equality of (10) and (11) is representing different types of expressions for Problem I. While (12), i.e., Problem II, is relatively varying from Problem I. Hence, (10) cannot transform into (11).

Multiobjective optimization is the mathematical problems that may concern more than one objective function's set to be optimized at once. Problem II relates to searching a transferring path $p \in P$ that can minimize $O_w(x)$ and $O_a(x)$ simultaneously and considered as a multiobjective mathematical problem. Next, we discuss the relationship between the solutions of Problem I and Problem II, established through following propositions with the proofs of each of them.

Definition 2 (Pareto Optimality): A solution $p \in P$ is supposed to be a Pareto optimal, with respect to all paths P between OD servers, if and only if there is no $p' \in P$ for which $w = O(p') = (o_1(p') \dots o_k(p'))$ dominates $a = O(p) = (o_1(p) \dots o_k(p))$.

The defined Pareto optimal in Definition 2 is interpreted as the entire decision variable space unless otherwise specified. In simple words, it can be explained that x' is Pareto optimal if there is no feasible vector x exist, which can decrease some criterion without an increase at the same time in at least one other criterion in minimization.

Definition 3 (Strong Pareto-Optimal Solution): $p^* \in P$ is a Strong Pareto-Optimal solution of Problem II, if and only if there cannot exist any other $p \in P$, such that $O_w(p) \leq O_w(p^*)$ and $O_a(p) \leq O_a(p^*)$.

Definition 4 (Weakly Pareto-Optimal Solution): $p^* \in P$ can be considered as a Strong Pareto-Optimal solution of Problem II, if and only if there cannot exist any other $p \in P$, such that $O_w(p) < O_w(p^*)$ and $O_a(p) < O_a(p^*)$.

Proposition 1: The solution related to Problem I is a weakly Pareto-Optimal solution of Problem II.

Proof 2: Let $p' \in P$ exist as a solution for Problem I. Let us assume that it as not a Weakly Pareto-Optimal solution of Problem II. For instance, there must exist a solution $p \in P$ as such $O_w(p) < O_w(p')$ and $O_a(p) < O_a(p')$. With the consideration of these assumptions, the weighted cost is set as $C_w \geq 0$ and $C_a \geq 0$ for at least one is larger than 0. Thus, $C_w O_w(p) + C_a O_a(p) < C_w O_w(p') + C_a O_a(p')$. ∎

There is inconsistency in terms of guessing that solution p' is the solution of Problem I or p' must be a weakly Pareto-optimal solution of Problem II.

Proposition 2: The result of Problem I is considered as a Strong Pareto optimal for Problem II, if the cost of the two subobjectives in Problem II are both positive, i.e., $C_w > 0$ and $C_a > 0$.

Proof 3: Let $p' \in P$ be the solution of Problem I as positive cost weights. Assuming that it is not a Strong Pareto-optimal solution. This signifies that an existence solution $p \in P$ such that $O_w(p) \leq O_w(p')$, $O_a(p) \leq O_a(p)$ and that $O_w(p) < O_w(p')$, $O_a(p) < O_a(p')$ holds for at least one. Since $C_w > 0$ and $C_a > 0$, we have $C_w O_w(p) + C_a O_a(p) < C_w O_w(p') + C_a O_a(p')$. While, this solution controverts the guess that p' is a result of Problem I specifies that solution p' must be strong Pareto optimal. ∎

Through the above findings, we can explain that the sum of delay cost and bandwidth weights in (10) that provides a better QoS solution in traffic steering for seamless service migration. Concretely, the method of cumulative delay and bandwidth cost finds a better transferring path that can minimize the delay and achieves higher bandwidth simultaneously. Furthermore, it ensures a strong Pareto optimality in (11) for the selected path. The scalability problem still exists in some situations.

III. Problem Formulation and Proposed Solutions

In vehicular communication, reliable mobility is reflected by ultralow latency. The current problem is a large-scale deployment of the MEC network that can never apply the traditional network routing techniques due to interdomain limitations [9]. Whereas, the proposed traffic steering for the MEC system [6], [12], [13] fails to minimize the time slot due to central control in a large-scale MEC network. Our goal is to minimize the time complexity to achieve QoS with a decentralized and fully optimized dynamic path selection framework for reliable mobility of vehicular application.

A. Scalability Problem

The scalability problem arises when the deployment of the MEC network expands to a large network [7], [9] that inherits several penalties.

1) The execution of the path selection method becomes intensely expensive (e.g., energy consumption and running time cost).
2) The SDN controller will take more time to collect information from so many nodes.
3) As the time cost becomes high, the efficiency of the results will be extensively reduced.
4) It is difficult to handle a large network as a traffic matrix, especially for the time-sensitive application process and migrations.

Observing 5G enabled MEC, we can find that large MEC is decentralized tractable with SDN and NFV [9], which helps to identify relevant important substructures in the MEC network. So, we identify the subsequent transformation.

1) Utilizing the static MEC network features that are additive and constant and cannot be changed in time after deployment.
2) A large MEC network can be partitioned or converted into a subnetwork block, based on unique properties of two types of nodes (routers and servers) in one network as described in Definition 5.
3) Only related nodes can be traversed during path searching with predefined and static MCNs as defined in Definition 6.
4) Both the above features need to be accessed only once at deployment time of the MEC network and helps to generate the prepossessed static results for dynamic path selection.

The collaborative MEC network framework [8] resolves this problem efficiently with its distributed computation power at each ES of network. With the above assumption, the network partition of the MEC network graph G can be defined as Definition 5;

Definition 5 (MEC Partitioned Boundary): A MEC network G is with a set of nodes $N = R \cup M$, where R represents the router and M is set of ESs (MEC servers) nodes. Let the interior network graph H be the subnetwork graph of G that can be obtained by moving the origin ES node as $m_o \in N$ and

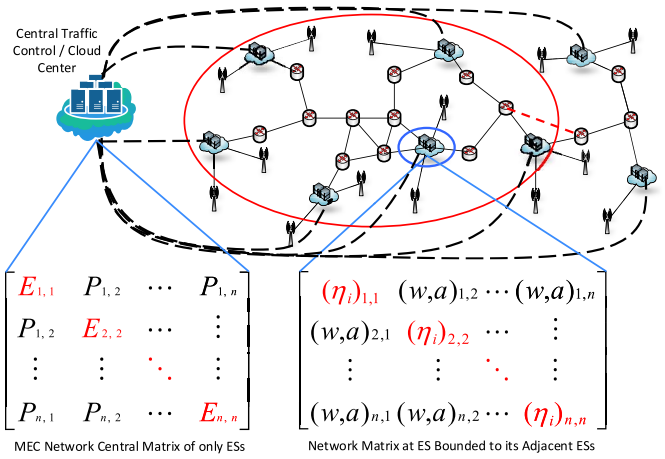

Fig. 3. Centralized and distributed control model of the MEC network with its central and interior network traffic matrices.

all its adjacent ES nodes ($[m_1, m_2, \ldots m_\alpha] \in N$) with their intermediary incidence links ($l \in L$) and routers ($r \in N$) from G, where α is the total number of adjacent ESs. With respect to $m_o \in M$, links $\forall l \in H$ and nodes $\forall m_\alpha \in H$ and router in between of servers ($\forall r \in H$) are called interior links and nodes, respectively. The remaining links ($l \notin H$) and nodes ($n \notin H$) are called exterior.

B. Problem Transformation

An ES ($m_o \in M$) is known as the origin ES that is configured to have its interfaces connected to its adjacent ESs ($m_\alpha \in H$) throughout the linked routers between them ($r \in H$). In simple, each adjacent ES works as an autonomous system boundary router [9]. What is more, this procedure needs to execute only once to create an adjoin ES's feature fusion block at each ES as an interior network matrix (INM) of H. In this scenario, origin ES ($m_o \in H$) keeps link metrics that can be advertised by the interior network nodes ($\forall n \in H;\ n_i \neq m_o$) in its INM, bounded to its adjacent ESs ($m_1, m_2 \ldots m_\alpha \in H;\ m_i \neq m_o$) throughout all incidence routers ($r \in H$) and links ($l \in H$). In short, an ES has only access to control and maintain the routing of its boundary area limited to its adjacent ESs. For instance, ES_1 creates a virtual adjoin boundary region, in Fig. 3, where the red circle area presenting a subnetwork is bounded to its adjacent ESs. Although, an INM of this bounded area operated by ES_1 is enlarged. A red dotted line extends the boundary if it exists, by connecting an interior router ($r \in H$) to the exterior router ($r \in G$) in the range of all next ESs.

A central traffic steering server also is identified connected to ESs $\forall M \in N$ through black dashed lines (in Fig. 3). Consequently, the whole scenario can be described as a two-layered system of traffic steering in 5G MEC: 1) decentralized traffic steering using the collaborative MEC control system and 2) centralized MEC network framework. The second layer works as a central traffic steering, which holds only the control matrix of all ESs only and their connectivity with each other. What is more, it can separate the load information of ESs from routers in the core network through a matrix-based centralized traffic attention network. However, the central traffic control with support of decentralized traffic steering would be discussed in our next research work to validate a flexible and collaborative MEC network.

Algorithm 1: ANBD Algorithm

Data: $[C_{i,j}]_{n\times n};\ n = |N|$; Source ES $m_o \in M \subset N$
Result: $[C^H_{i,j}]_{n\times n,\ n=|H|}$

1 $Stack \leftarrow m_o$
2 **while** *Stack not empty* **do**
3 $element \leftarrow$ Pop Stack
4 **if** $element \notin H$ **then**
5 **if** *element ID has character 'E'* ***and*** $element \neq m_o$ **then**
6 $H \leftarrow element$
7 Return to line 3
8 **else**
9 Stack $\leftarrow \forall$ Neighbors $\in element$
10 $H \leftarrow element$
11 $Predecessor[\forall$ Neighbors$] \leftarrow element$
12 $([C^H_{i,j}]_{n\times n};\ n = |H|)$, for all C_i and $C_j \leftarrow n_i \in H$
13 **For all** $C_{i,j} \leftarrow l_{i,j} \in H$ Where; $Predecessor[i] = j$
14 **Return** $[C^H_{i,j}]_{n\times n}$

C. Scalability Solution Through ANBD

With the consideration of Definition 5, we propose an ANBD algorithm that bounds a large MEC network traffic routing as a partitioned interior subnetwork ($H \subset G$). The ANBD framework runs at each ES ($\forall m \in M$) and creates its own block of the interior network based on its adjacent ES ($m_1 \ldots m_\alpha \in m_o$) and controls its outbound traffic, to a large extent. The ANBD constructs an adjoin boundary matrix $[C^H_{i,j}]_{n\times n};\ n = |H|$. As defined in Definition 5, the nodes of the MEC network ($\forall n \in N$) can be separated as $N = M \cup R$, where $M \subset N = \{m_1, m_2, \ldots, m_k\}$ is a set of MEC servers and $R \subset N = \{r_1, r_2, \ldots, r_n\}$ is the set of routers.

The boundary discovery algorithm ANBD (Algorithm 1) works with BFS searching technique, which takes MEC network G as an adjacency list and the origin ES ($m_o \in G$) as input. Throughout searching, every *element* node $n \in N$ in stack (in line 2), if the *element* is an ES, then it is inserted in interior network node set H directly without searching its neighboring nodes (as lines 4–7). Otherwise, all the nodes are considered as router nodes. Lines 9–11 explore the neighboring nodes of all the remaining nodes *element*, push them into stack if not visited, and assigns *element* as *Predecessor* for all neighboring nodes to establish the links between n_i and n_j. Lines 12 and 14 construct a subnetwork traffic matrix ($[C^H_{i,j}]_{n\times n};\ n = |H|$) with the help of interior network nodes ($\forall n \in H$) and create the link connectivity explored by *Predecessor*.

Algorithm 2: MCNB Algorithm

Data: $[C^H_{i,j}]_{n\times n}$; Source ES $m_o \in M \subset N$
Result: Sets of must connected nodes where; number of sets = number of adjoin ESs.

```
mcnb ← dictionary initialization
adjoinESs ← ∀m ∈ M ∈ H where; m ≠ m_o
foreach ES ∈ adjoinESs do
    AllPath ← AllPath(H, m_o, m_d)
    foreach node ∈ AllPath do
        if node ∉ set P then
            set P ← node
        mcnb[ES] ← set P
return mcnb
// Function for finding all possible linked nodes!
def AllPath(Graph, Start, End, Nodes = [ ]):
    Nodes ← Names + Start
    if Start is End then
        return Nodes
    Check ∀ Neighbors ∈ start
    if neighbor ≠ End and ∉ Nodes then
        Paths ← AllPath(Graph, neighbor, End, Nodes)
    return Paths
```

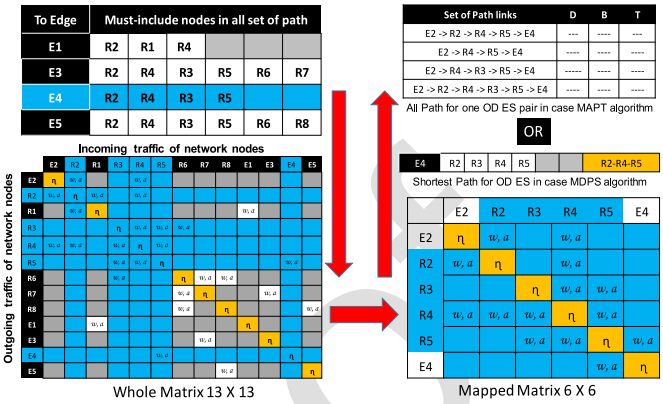

Fig. 4. Predefined memory-based MCNB set and MCN-based matrix minimization process steps for the path selection procedure within the related node.

D. Fine-Grained Solution Through MCNB

Moreover, the feature observation of the MEC network model motivates the needs for a finer-grained characterization of interior network H in terms of the identifiability of MCNs between OD pair of ESs. Dynamic shortest path finding throughout MCN can be expressed as a general case of the well-known traveling salesman problem, which is also known as NP-complete [21]. Instead of traversing all nodes as in the original traveling salesman problem, this problem involves to visit only a subset of predefined MCN set between OD ESs. In particular, it is possible to easily trace out all connected nodes and link paths, where all MCN sets must satisfy the property of linearly independent paths (as in Definition 6). With this assumption, a path-finding process based on MCN for all subnetworks ($\forall H \in G$) can be defined in Definition 6.

Considering Definition 6, we obtain an MCN set, using Algorithm 2, of the possible linked nodes to find all linearly independent connectivity for every adjacent ES ($\forall m_\alpha \in H$) as destination, from origin ES ($m_o \in H$). Algorithm 2 is also only one time required to be executed and generate a static immutable MCNB of must linked node sets. As shown in Fig. 4, the MCNB sets $S^{\text{mcn}} = \{s_{m_1}, s_{m_2}, \ldots, s_{m_\alpha}\}$ contains MCN for all adjacent ESs ($\forall m_\alpha \in H$) in each set. Before the execution of the path-finding process, a set of nodes is selected by the requested destination ES ($m_d \in H$) (e.g., ES_2, $\text{ES}_3, \ldots, \text{ES}_k$). Throughout this procedure, a reduced traffic matrix $[C^R_{i,j}]_{n\times n}$; $n = |s_{m_i}| \in S^{\text{mcn}}$ of only the related nodes between OD ESs pairs can be accessed, instead of nonrelated nodes in whole interior network H. Fig. 4 represents the reduced matrix of 6×6 instead of full traffic matrix of 20×20 nodes to minimize the searching time of nonrelated nodes. Algorithm 2 representing MCNB procedure in two parts. First, the main idea is to find all linearly independent paths between origin ES ($m_o \in H$) to all adjoin ESs ($m_\alpha \in H$), with a recursive function, named *AllPaths*. Second, it creates an MCNB table of sets of independent but unique nodes.

Definition 6 (Linearly Independent Paths): A path p_i can be linearly independent if and only if it has at least one new link ($l_{i,j} \in H$) that is not exist in any other path set ($p_1, p_2, \ldots, p_k$; $k \neq i$). In other words, if a path set p_i has at least one new node ($n_i \in H$) as compared to other linearly independent paths, then this path is also considered as a linearly independent path, because any single path p_i containing at least one unique node automatically indicates that it has a unique link. A path p_i if showing as a subpath of another linearly independent path P_k and then it will not considered as a linearly independent path set.

The thing that should be remembered at the end of this discussion is that both ANBD and MCNB algorithms need to be executed only once at MEC deployment and generate an immutable results. The dynamic path selection algorithms, discussed in the next section, is executed based on these static results dynamically to minimize the processing time slot.

IV. Proposed Dynamic Path Selection

In this section, we discuss our proposed dynamic path selection algorithms, for seamless service migration, based on both ANBD and MCNB devised solutions, discussed in the previous section.

A. Shortest Path Selection Approach Through MDSPS

Generally, a very important constraint for seamless service migration is transferring time. If the transferring time is large, it will have a bad impact on QoS [2]. In some kinds of mobile-based applications, if the time of migration achieves as less

than 0.1 s, then the service continuity remains imperceptible. While, if the response time of migration is in the range of [0.1; 1] s, it becomes difficult to maintain the user's continuity, as well as larger than 10 s, it will be impossible to remain appropriately connected [11], [22]. Therefore, we introduce the notion of the time threshold $\overline{T}$ for QoS and seamless migration of services. If the transferring time is shown less than this threshold time, the migration of service will be transparent for mobile users during the movement from one ES to another. It is clear that the fixed threshold time value is not applicable across all application types [7]. The MDSPS algorithm consists of two parts presented in Algorithm 3. The first part generates a minimized matrix $[C^R_{i,j}]_{n\times n}$ of only MCNs from INM $[C^H_{i,j}]_{n\times n} \in H$. The second part finds out a shortest best path for data transferring based on the enhanced Dijkstra algorithm with consideration 7.

The main target is to find a new shortest and best transferring path p, in each cycle, until these paths together transfer all data K of running service in time T. For minimizing the cost of searching time we introduce a related only matrix of nodes called reduced matrix by predefined MCN set (Lines 3–5). Based on our enhance Dijkstra structure, the selected link $l_{i,j} \in p$ ensures the higher available bandwidth and minimum sum of delay cost from line 15 to 17. These objective functions are formed as (6) and (7) and guarantee the fully optimal solution as proved in Section II-C. Line 24–27 arrange path node with the help of *predecessor* and then simply calculate the delay sum of path $D_p^{o\to d}$ from total distance of m_o and bandwidth of path $B_p^{o\to d}$ by (6) (as Lines 28 and 29). Line 30, operations to find the initial minimum time period in which all data can be transferred. This is assured by Proposition 3. Hence, it is checked that if data volume K is less than bandwidth of selected path $B_p^{o\to d}$, the path p and threshold time T will return. On the other hand, in lines 33–35, data is relaxed and sent in partition as per window size of path at each time slot.

Proposition 3: $T = Q/B_p^{o\to d}$ is a minimum time cost period needed for transferring K sized data, depending on the set containing linearly independent connected links and nodes denoted as p, where $B_p^{o\to d}$ means the bandwidth of transferring path p.

Proof 1: Given an arbitrary link in path p, let it be $(l_{i,j} \in p)$, the transferring time on it $t_{i,j}$ is less than T, i.e., $t_{i,j} < T$. Using the pigeonhole principle example [18], there exists at least one transferring link $l \in p$ with transferring time $t_l \geq ([Q - a_{i,j}.t_{i,j}]/[B^p - a_{i,j}]) > ([Q - a_{i,j}T_l]/[B^p - a_{i,j}]) = T$. This signifies that there will never be the transferring time of all links $(\forall l \in p)$ is less than T. So, T becomes the minimum time slot or time cost. ■

The MDSPS acts only as the shortest but the shortest path selection (SPS) algorithm in one time slot. It can increase time complexity during the management of multiple requests. If more than one service call migrates operations simultaneously, the algorithm will not response in the ideal time period. The question arises that how to process the multiple requests for the best path between two ESs even in a subnetwork H. This is where the multipath searching (MPS)-based MDMPS algorithm comes in.

Algorithm 3: MDSPS Algorithm

Input: $[C^H_{i,j}]_{n\times n}$ as INM graph, K data size to be transferred, Source ES $m_o \in M \subset H$, target ES $m_d \in M \subset H$ and $(mcnp = MCNB[m_d])$ predefined must linked node set

Output: A best shortest path p with shortest delay, higher available bandwidth as Eq. (6) and estimated time T for seamless service migration

1 */First part: create a reduced/minimized matrix. /*
2 INITALIZE $[C^R_{i,j}]_{n\times n}$; $n = |mcn|$
3 **foreach** *i in mcn* **do**
4 **foreach** *j in mcn* **do**
5 $[C^R_{i,j}]_{n\times n} \leftarrow [C^H_{i,j}]_{n\times n}$
6 **For all** $C^R_{i,j} \leftarrow$ set *dist*[ ] as ∞ where; $C_{i,j} \neq m_o$
7 *Queue* $\leftarrow 0 \in m_o$ */ set priority queue /*
8 **while** *Queue not empty* **do**
9 *item*, *dist*(*item*) $\leftarrow$ Pop *Queue*
10 **if** *item not visited* **then**
11 **foreach** *Neighbor as* $j \in$ *item as* i **do**
12 **if** $j \neq i$ **then**
13 **if** $i = m_o$ **then**
14 $cost(t) \leftarrow \sum_{i\to j} w_{i,j} + a^{-1}_{i,j} + dist(i,j)$
15 **else**
16 $cost(t) \leftarrow \sum_{i\to j} w_{i,j} + a^{-1}_{i,j} + dist(i,j) + \eta_i$
17 **if** $cost < dist(i,j)$ **then**
18 (i)$dist(i,j) \leftarrow cost(t)$
19 (ii)$Queue \leftarrow j \in cost(t)$
20 (iii)$Predecessor(j) \leftarrow i$
21 mark *item* as visited
22 */ Compute Path with delay and bandwidth values /*
23 *temp* $\leftarrow m_d$
24 **while** $m_o \notin p$ **do**
25 $p \leftarrow predecessor[temp]$
26 $temp \leftarrow predecessor[temp]$
27 COMPUTE $D_p(t) \leftarrow dist(m_d)$
28 COMPUTE $B_p(t) = \min_{l_{i,j}\in p} a_{i,j}$ */ Using Eq.* (5) /*
29 COMPUTE $T_p \leftarrow K/B_p(t)$ */ practical transferring time of path p /*
30 $P \leftarrow P \cup \{p,\ T_p\}$
31 **if** $\overline{T} > T$ **then**
32 **return** P
33 **else**
34 UPDATE $K \leftarrow K - (B_p \times \overline{T}_p)$
35 Go to line 3

B. MPS Approach Through MDMPS

In this section, we propose an MDMPS algorithm, where every path set $p_i \in P$ constructs its path matrices, by the

Algorithm 4: MDMPS Algorithm

Input: $[C^H_{i,j}]_{n\times n}$ as INM graph, K data size to be transferred, *All_Path_Dic* having all path sets between $m_o \rightarrow \forall m_a \in H$; Origin ES $m_o \in H$ and Destination ES $m_d \in H$

Output: Multi-path set P with having total delay as Eq. (7) and available bandwidth as Eq. (6) at each path $p_i \in P$

1 $m_d\_Paths \leftarrow All\_Path\_Dic\ [m_d]$
2 **while** m_d_Paths *not Empty* **do**
3 **foreach** $p_i \in m_d\_Paths$ **do**
4 **foreach** *node* $(i, i+1) \in p_i$ **do**
5 **if** $i = m_o$ **then**
6 SET $D_{p_i} \leftarrow \sum w_{i,j} \in C_{i,j}$; where $j = (i+1)$
7 **else**
8 SET $D_{p_i} \leftarrow \sum w_{i,j} + \eta_i \in C_{i,j}$
9 SET $B_{p_i} \leftarrow a^{min}_{i,j} \in C_{i,j}$
10 POP $p_i \in m_d\_Paths$; with $D^{min}_{p_i}$
11 COMPUTE $T_{p_i} \leftarrow K/B_{p_i}$
12 $P \leftarrow P \cup \{p_i,\ T_{p_i}\}$
13 **if** $\overline{T} > T_{p_i}$ **then**
14 **return** P
15 **else**
16 **if** $\sum_{p\in P} T_{p_i} < \overline{T}$ **then**
17 UPDATE $K \leftarrow K - (B_{p_i} \times \overline{T})$
18 Go to line 10
19 **else**
20 Go to line 1

evaluation of only related row–column connection in the traffic matrix. The dynamic searching environment through INM guarantees the lowest execution time cost than the online network link accessing techniques [23]. As well as, there is no need to compare the cumulative link cost and bandwidth of each link as in MDSPS modules by simply single access and INM element values with predefined path set nodes. In short, Algorithm 4 explains the main and simple feature of the MDMPS algorithm, where only predefined and identified nodes in the selected path set can be traverse in INM.

We take into account the `AllPath` function only (in Algorithm 2) instead of the whole MCNB algorithm, for constructing the all path dictionary (`All_Path_Dict`), where every ES name is assigned as a key, which specifies its all connected path sets. The `AllPath` function is called for all adjoin ESs ($\forall m_a \in H$) and returns a dictionary of path sets (`All_Path_Dict`), for every adjoin ES from origin ES. This procedure is also only one time executed and, as a result, `All_Path_Dict` is created as the predefined path sets.

In Algorithm 4, the matrix-based all path table (MAPT) takes INM matrix $[C^H_{i,j}]_{n\times n}$ as input graph with origin ES m_o, destination ES m_d, and all path dictionary `All_Path_Dict`. A set of paths, between OD ESs, from $m_d \in All\_Path\_Dict$, is selected by the key name of the destination ES m_d. Only required node's values, from INM, are accessed and computed with the node's name in the selected path set $p_i \in m_d$, from lines 5 to 9). The delay $w(t) \in C_{i,j}$ and available bandwidth $a_{i,j}) \in C_{i,j}$, where $i, j = i, i+1 \in p_i$ is set as the total delay $(\sum D(t) \in p_i)$ and minimum bandwidth $(a^{\min}_{i,j} \in p_i)$ and are updated to identified single path $p_i \in m_d$. Line 10 selects a path with the lowest delay cost and computes a practical transferring time T_{p_i} that is assigned to its path in multipath set P in lines 11 and 12. The remaining procedure checks transferring time $T_{p_i} \in P$; if it can transfer all data K and then return it. Otherwise, select next path from $p_{i+1} \in m_d\_Paths$ in multipath set p, for sending data in parallel on multipaths.

The most important thing should be discussed here that the extra advantage of the MDMPS algorithm with the `AllPath` function in Algorithm 2 is that it can construct a static table of all connected node paths toward all adjoin ESs $\forall m_a \in H$. However, these paths, referred to as static paths, include definite connected nodes only and their link connectivity. By using the MDMPS algorithm for all adjacent ESs, it can be used as a routing table (e.g., BGRP) [9]; but in the MDMPS model, there is no need to create protocol tables separately by all nodes ($n \in N$). With the beneficial effects of the bounded and partitioned MEC network, all path routing tables can be built in a minimum time triggering time. This efficiency to handle routing paths in the subnetwork can be executed separately in a specified time and use the updated path matrices to avoid the whole process of SPS or MPS at each request. The evaluation results also prove that generating and using the routing table by MDMPS, all traffic routing and service migration achieve state-of-the-art results in time sensitivity.

V. Result Evaluation

We know that the dynamic path selection techniques with the interior MEC boundary framework can theoretically give np-complete optimal solutions in both whole and partitioned MEC network planes, but the experimental tests will validate the proposed methods.

A. Setup

To achieve reliable results, both algorithms are performed 100 times at different node-sized portioned MEC networks. The whole MEC network is considered with a total of 100 nodes, including 30 ESs and 70 routers. Due to the values of link matrices handled locally and offline as memory-based traffic matrix and to avoid the complexity, all values are generated randomly, where normalized delay $(w_{i,j})$ of link $(l_{i,j})$ is between [0.0, 1.0] and capacity $(c_{i,j})$ is limited between [0.5, 1.0] Gb/s. Although the available bandwidth of link $(a_{i,j})$ can be calculated in simple through 3, to reduce the influence of the simulation environment, extreme results are rejected, and the average of remaining results is calculated. In addition, we list each of our proposed algorithm with the application of four benchmark schemes and their relations, and present the results with respect to each algorithm, as follows.

1) *Bounded (B):* The ANBD algorithm generates a partition-based immutable controlled matrix on the current Server, which is represented by the Bounded (*B*) benchmark. By using the ANBD algorithm at a MEC server, the current server only can see its adjacent Servers only.
2) *Mapped (M):* To avoid the nonrelated servers and routers during path searching, we introduce an MCNB algorithm that creates immutable lists of MCNs and is known by the Mapped (*M*) benchmark, where each node list is represented by its adjacent MEC server.
3) *Bounded and Mapped (BM):* A distributed traffic steering works under both the above benchmarks as (BM).
4) *No Filter:* Here, the MEC network is considered as a whole network and will not apply any of the partitioned and mapped techniques as discussed above.

The benchmarks were conducted on both MDSPS and MDMPS algorithms. These network algorithms are not only technically design level but are also practically applicable level using the result of the benchmark test.

We have a set of OD ESs pairs, for all experiments, and calculate the final result as the mean value from all evaluation metrics. To provide clear distinctions and visualization for the plots based on four benchmark schemes, the plots are presented in the same way. In fact, we have four benchmark schemes for both algorithms in our research; as for the SPS module, it can be considered as SPS-BM, SPS-M, SPS-B, and SPS, as well as for MPS module, it is MPS-BM, MPS-M, MPS-B, and MPS. We work with three evaluation metrics as follows, to estimate the performance of the proposed MDSPS and MDMPS algorithms.

1) *Computational Time:* The time required by the algorithms to compute the desired path selection on both MDSPS and MDMPS algorithms.
2) *Transferring Time:* It is the required time to transfer service data between the OD pair of ES and is also related to the data volume to be transferred and available bandwidth.
3) *Node Traversing:* The range of the maximum number of nodes to be searched in the MDSPS algorithm. It is related to the benchmark schemes for node searching.
4) *Distance of OD ES:* The evaluation of computational time based on the distance as the minimum number of nodes between OD ESs in one path.

Good performance is considered based on the lower value of the above three evaluation.

B. Performance Analysis

We compare the MDSPS and MDMPS algorithms based on the above-discussed benchmark schemes. As well as, the importance of decentralized traffic steering is compared to the centralized algorithm, named PLP, has proposed in [7], which gives the best results. The important similarity that needs to be compared is the use of the elliptic region with a filtering version of PLP as PLP/F. Note that we start with a large region size of the network than that is used in PLP and PlP/F that also not provide fully optimal path selection with the lowest delay and higher bandwidth. On the other hand, we already prove with strong Pareto-optimal analysis that our proposed framework gives fully optimal results between delay and bandwidth matrices, in Section II-C.

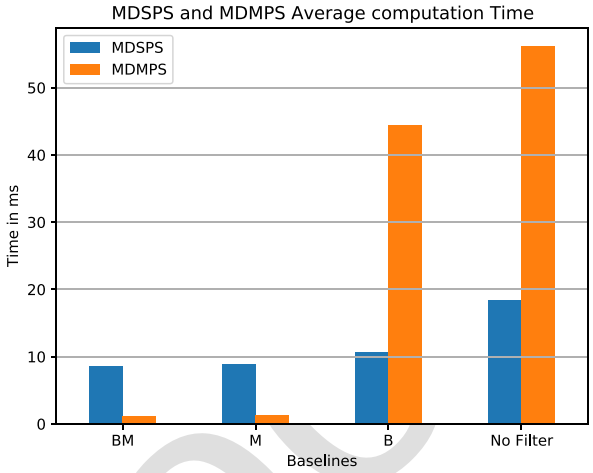

Fig. 5. Average experimental results of both MDSPS and MDMPS with four benchmark schemes.

Fig. 5 represents the average running time of the MDSPS algorithm and total computation with selected but different data sizes as managed in [5] and [24]. It is closely seen that the computational time of SPS-BM and SPS-M is 8.62 and 8.84 ms, respectively, which is very low running time cost than SPS-B and only SPS (as a whole MEC network). The total time of transferring the data is relevant to the required bandwidth for data to be transferred and computational time of algorithm. Obviously, the MDSPS algorithm has the best case time complexity of $O(2N+N^2)$; whereas, the time complexity of the Dijkstra algorithm is $O(N^2)$. From Fig. 5, we observe that the QoS of the migration process is highly affected by the computational time in the MDSPS algorithm, where the minimum or only related node traversing (as with BM and M benchmark schemes) has become more important. Hence, more than one request for path searching will increase the transferring time too and affects threshold time.

We pursue a similar evaluation, but with better behavior than MDSPS is observed by the MDMPS algorithm's experimental results. In fact, a better pattern of plots is acquired, which is obvious in Fig. 5. Hence, we conclude that the MDMPS algorithm performs extremely lowest computational time in MPS-BM and MPS-M with 1.14 and 1.23 ms, respectively. This is caused by the best time complexity of $O(|P|\times(N-1))$, where $|P|$ is the number of linearly independent path sets. It is closely seen that path searching with MPS-BM and MPS-M, all independent paths are selected for only one destination ES even then it achieves the extremely lowest computational time than SPS-BM and SPS-M. Whereas, the MPS-B and MPS generate an all-path table for each adjacent ES ($\forall m_\alpha \in H$) as an independent path block. So, it is observed that a required path can be accessed from the table, instead of whole computation at every request for service data transfer.

In Fig. 6, we perform a random data size with threshold time $\overline{T}$ of 1s to check the available transferring time

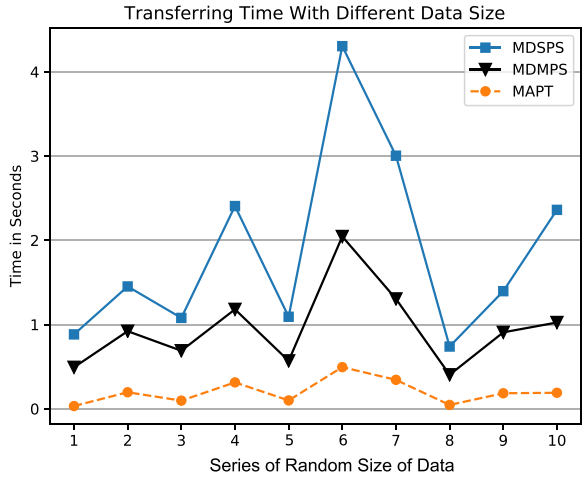

Fig. 6. Experimental results of random different data sizes with the best case of MDSPS and MDMPS algorithms.

at both MDSPS and MDMPS algorithms. We also consider and define MSP-B and MSP conditions as a routing table form the origin to all adjoin ESs in both subnetwork and whole network. We named it MAPT and can work as a routing table as in the boundary gateway protocol (BGP) [9]. With boundary-based MEC networking, it is easy to update the routing table with 1-s interval time without any network congestion. Simply, it chooses the best updated path without executing the whole MDMPS algorithm. The difference between MDSPS and MDMPS-based transferring time is equally different if data transferring time becomes upto 1 s (i.e., 1, 3, 5, 8 levels). Although, larger threshold time differs larger between MDSPS and MDMPS. The main difference is generated by the single-path searching technique of MDSPS than MDMPS and MDSPS every time adds its computation time more than 1-s time threshold. Whereas, MDMPS works on multiple paths at once in simple access of each element in the matrix related to path nodes. The higher quality results are given by path selection as the all-path routing table with MPS-B and MPS, simultaneously, help to minimize the time threshold for live and seamless service migration. It can be noticed easily as the MAPT line graph in Fig. 6 that the additive time value of path selection in the total transferring time is very low and can perform a better QoS in seamless service migration in the vehicular MEC network. In short, we generate best computational time and transferring time with the multipath technique instead of single shortest path, as well as better performance than that is discussed in [7].

Fig. 7 exemplifies the execution time affected by the distance between the selected OD ES pair. The distance is considered as one path with minimum MCN nodes between OD ES. We evaluate both of our algorithms with four discussed benchmark schemes for better understanding. In Fig. 7(a) and 7(b), the increment within the minimum MCN nodes' path between OD ES escalates the computational time in SPS than the MPS, where matrix mapping to MCN nodes adds its computational time. This is caused by the computation of the MCNB algorithm by increasing the distance between OD ES. With simple computation of the MDMPS algorithm, we achieve better computational time in both BM and only M benchmark schemes with must related nodes only. Whereas, the MDSPS algorithm shows more computational time due to comparative calculation and updating from node to node $n_i \rightarrow n_j$. In Fig. 7(c) and 7(d), without the MCNB algorithm, MDMPS computational time grows exponentially than the MDSPS framework caused by all nonrelated nodes traversing. With the evaluation for comparisons of both SPS and MPS modules based on all benchmark schemes, it is assumed that both of our MDSPS and MDMPS algorithms provide the best results in different scenarios. Although if the node distance is less than 20 nodes, then the MDMPS algorithm can work well than an SPS (MDSPS) framework with predefined and immutable MCN path nodes.

C. Parameter Analysis

The graph characteristics in Fig. 8 shows the rate of change of nodes traversing in multi iterations with different baselines. It is very clear that MCNB-based node mapping (SPS-BM and SPS-M) generates better results by only related node traversing and avoiding the unrelated nodes and paths. This is why SPS-BM and SPS-M produce same results that overlapp both lines. In Fig. 8(a), the graph shows the rate of node traversing with 20-node partitioned/bounded network within the MEC network of 100 nodes. Fig. 8(b) with the bounded node of 30, Fig. 8(c) resultant with a 40-node cluster of the MEC network, and Fig. 8(d) are shown with 50-node bonded network INM. It should be noted here that all simulation results are triggered and generated at 100% rate of updated link values at every iteration or time interval. Although, the MDMPS framework always traverses all nodes between OD ESs pair in every benchmark schemes; whereas, the computational complexity always matter in MDMPS than that in MDSPS, which has been discussed.

The effects of the bounded region size implementation with all benchmark schemes, in detail, is shown in Table II. We set region sizes with the number of nodes as 20, 30, 40, and 50 nodes in a MEC network, respectively. All experiments are executed at both MDSPS and MDMPS, with different size of data to obtain the tentative results. The whole network consideration without bounded and mapping still performs badly in both SPS and MPS benchmark schemes on running time. The MDMPS algorithm specifically performs better caused by its simplest cumulative procedure of link and nodes matrices. As the number of node increases, the evaluation performance between bounded and no filter approaches one another. As a result, computation will increase to find the best transferring scheme. As the same, the time-changing ratio with small a difference than MDSPS is observed with experimental results, the MDMPS algorithm performs with the same region size of the MEC network. However, the MDMPS framework achieves extremely better results with ANBD and MCNB than the MDSPS algorithm.

The evaluations and tables significantly indicate that our MDSPS algorithm is efficient in terms of time involvement and accuracy than the existing shortest path algorithms [7]. As well as, the proposed MDMPS is more efficient in terms

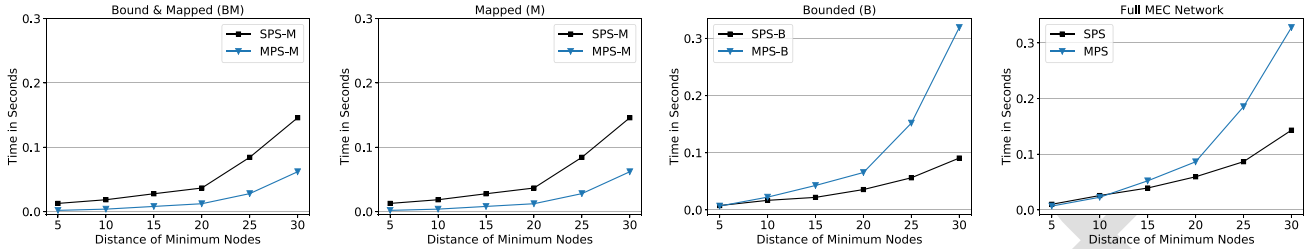

Fig. 7. Computational time is measured based on the distance between OD ES. AQ3

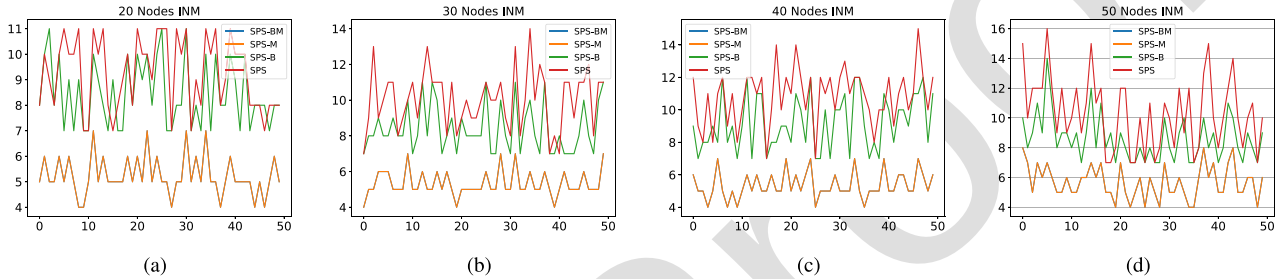

Fig. 8. Experiment in 50 iterations of traversing the total number of nodes in each benchmark schemes of the MDSPS algorithm.

TABLE II
EFFECTS OF REGION SIZE

Metrices	MDSPS Computation Time (ms)				MDMPS Computation Time (ms)			
	20 Nodes	30 Nodes	40 Nodes	50 Nodes	20 Nodes	30 Nodes	40 Nodes	50 Nodes
BM	6.1844	7.4436	10.4247	10.44	0.9812	1.1971	1.1929	1.1998
M	8.9438	7.6051	8.1563	10.6492	1.181	1.2334	1.2397	1.2712
B	8.0208	9.2407	10.463	12.6919	11.9526	33.908	58.6736	73.0629
No Filter	8.8201	8.4422	8.9437	28.0969	12.3074	35.7549	62.0959	114.0885

of multipath set computation. Moreover, in the case of large size of MEC network graphs, ANBD and MCNB modules provide efficient bounded network distribution. In contrast, our MEC network partitioning approach is unique and provides better results for both dynamic and static path searching. It is observed that our whole proposed research is also very high in the sense of dynamic path searching in a large MEC network, with our algorithms performing well almost every time.

VI. Related Work

The standardization for 5G MEC is in the emerging era and there is no physical deployment work on live service migration, even on traffic steering. A very few research work is done on the dynamic path selection and traffic steering system.

MEC-based centralized traffic steering brings together in [6], [12], and [13] as cloud-based traffic monitoring, in which the researcher applies deep learning and machine learning (MDP, cognitive computing etc.) methods. Most of them have achieved impressive results in path finding but cannot comprise the required time complexity. Work in [25] proposed a jointly optimized method for content delivery using the Markov decision framework in the vehicular edge computing. However, the minimum time slot still remains challenging due to the centralized control system in a large MEC network. When we analyze the scalability of a large network, the distributed approach of the portioned SDN network [9] becomes important in 5G MEC. Our MDMPS can be considered a little bit similar to multipath TCP (MPTCP) [26]. While, MPTCP is a general solution for robustness and accuracy, rather than being customized with 5G MEC. Similarly, Xu *et al.* [7] proposed, very well, a centralized dynamic traffic steering for dynamic path selection. This research handles the large network scalability problem with a limited elliptic region size. Hence, the result shows that the optimal time complexity is still challenging. Kagami *et al.* [19] discussed the importance of bandwidth in network path connection with the cost of time in service migration which become the base of seamless service migration. Whereas, our ideology is that the dynamic path selection approach can be taken into account as static if the all-path routing table will consider for all adjacent ESs.

VII. Conclusion

This research focused on the minimize time slot for transferring path during vehicle mobility. This complement the flexibility through the reorganization of the MEC servers and routers separately, especially when more ESs join the vehicular edge computing. We proposed a distributed traffic steering model in a large-scale MEC network. The proposed

benchmark algorithms execute through the distributed network matrix bounded to its region. The distributed control feature efficiently solves the scalability problem of a large MEC system for dynamic path selection complexity. The benchmarks were conducted both single path selection and MPS algorithms. These network algorithms are not only technically design level but are also practically applicable level using the result of the benchmark test. We outlined our conceptual analysis and interpret experiment results with and without benchmark scheme algorithms. The experimental work manifested that our proposed model and algorithms can help to minimize time slot for seamless service migration, with a distributed control system. Accordingly, we will research how these inherent problems can be resolved with both proposed distributed and centralized technology and blockchain smart contract of collaborative MEC network.

REFERENCES

[1] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, Oct. 2018.

[2] S. Kekki *et al.*, "MEC in 5G networks," ETSI, Sophia Antipolis, France, White Paper, 2018.

[3] M. R. Anawar, S. Wang, M. A. Zia, A. K. Jadoon, U. Akram, and S. Raza, "Fog computing: An overview of big IoT data analytics," *Wireless Commun. Mobile Comput.*, vol. 2018, May 2018, Art. no. 7157192.

[4] Z. Ning *et al.*, "Joint computing and caching in 5G-envisioned Internet of Vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 5, 2020, doi: 10.1109/TITS.2020.2970276.

[5] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[6] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, Jun. 2019.

[7] J. Xu, X. Ma, A. Zhou, Q. Duan, and S. Wang, "Path selection for seamless service migration in vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9040–9049, Sep. 2020.

[8] T. Jin, W. Zheng, X. Wen, X. Chen, and L. Wang, "Optimization of computation resource for container-based multi-mec collaboration system," in *Proc. IEEE 30th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2019, pp. 1–7.

[9] M. Caria, A. Jukan, and M. Hoffmann, "SDN partitioning: A centralized control plane for distributed routing protocols," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 381–393, Sep. 2016.

[10] R. Alvizu *et al.*, "Comprehensive survey on T-SDN: Software-defined networking for transport networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2232–2283, 4th Quart., 2017.

[11] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, Jul. 2017.

[12] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A dynamic service migration mechanism in edge cognitive computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–15, 2019.

[13] 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," *Comput. Netw.*, vol. 108, pp. 357–370, Oct. 2016.

[14] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, May 2019.

[15] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 38–43, Mar. 2017.

[16] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *Proc. IEEE Global Telecommun. Conf.*, vol. 3, 2004, pp. 1629–1634.

[17] P. Megyesi, A. Botta, G. Aceto, A. Pescapè, and S. Molnár, "Available bandwidth measurement in software defined networks," in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, 2016, pp. 651–657.

[18] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 39–44.

[19] N. S. Kagami, R. I. T. da Costa Filho, and L. P. Gaspary, "CAPEST: Offloading network capacity and available bandwidth estimation to programmable data planes," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 175–189, Mar. 2020.

[20] A. Riedl, "A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics," in *Proc. IEEE Workshop IP Oper. Manag.*, 2002, pp. 166–170.

[21] B. H. Arabi, "Solving NP-complete problems using genetic algorithms," in *Proc. IEEE UKSim-AMSS 18th Int. Conf. Comput. Model. Simulat. (UKSim)*, 2016, pp. 43–48.

[22] S. Raza *et al.*, "An efficient task offloading scheme in vehicular edge computing," *J. Cloud Comput.*, vol. 9, pp. 1–14, Jun. 2020.

[23] D. Li, C. Xing, G. Zhang, H. Cao, and B. Xu, "An online dynamic traffic matrix completion method in software defined networks," *Comput. Commun.*, vol. 145, pp. 43–53, Sep. 2019.

[24] Microsoft. *How Much Bandwidth Does Skype Need?* Accessed: Nov. 1, 2020. [Online]. Available: https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need

[25] 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 Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.

[26] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. NSDI*, vol. 11, 2011, p. 8.

Muhammad Rizwan Anwar received the MCS degree in computer sciences from Punjab University of Lahore, Lahore, Pakistan, in 2015, and the M.S. degree in computer sciences from Virtual University of Pakistan, Lahore, in 2017.

He is currently a Ph.D. Researcher with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. He has research interests and focus on multiaccess edge computing, big data, AI, and dynamic collaborative computing.

Shangguang Wang (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2011.

He is currently a Professor with the School of Computing, Beijing University of Posts and Telecommunications, where he is a Vice-Director of the State Key Laboratory of Networking and Switching Technology. He has published more than 150 papers. His research interests include service computing, cloud computing, and mobile-edge computing.

Prof. Wang has served as a General Chair or a TPC Chair for IEEE EDGE 2020, IEEE CLOUD 2020, IEEE SAGC 2020, IEEE EDGE 2018, and IEEE ICFCE 2017, and a Vice-Chair for IEEE Technical Committee on Services Computing from 2015 to 2018. He is currently serving as an Executive Vice-Chair for IEEE Technical Committee on Services Computing and a Vice-Chair for IEEE Technical Committee on Cloud Computing.

Muhammad Faisal Akram received the B.S. degree in computer science from Allama Iqbal Open University, Islamabad, Pakistan, in 2006, and the M.S. degree in computer science from the University of Agriculture Faisalabad, Faisalabad, Pakistan, in 2010.

He is currently a Ph.D. researcher with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include mobile-edge computing, service computing, and 5G tactile Internet.

Salman Raza received the MCS degree with a distinction (Gold Medal) in computer science from Bahauddin Zakariya University, Multan, Pakistan, in 2009, and the M.S. degree in computer science from Government College University Faisalabad, Faisalabad, Pakistan, in 2014. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.

His current research interests include mobile-edge computing, vehicular networks, task offloading, and machine learning.

Shahid Mahmood received the B.S.Eng. degree in 2017 and the M.S. degree from Beijing University of Post and Telecommunications, Beijing, China, in 2020, where he is currently pursuing the Ph.D. degree. AQ5

His current research interests include edge computing and signal measurement.