EdgeChain: Blockchain-based Multi-vendor Mobile Edge Application Placement

He Zhu
Systems and Computer Engineering
Carleton University
Ottawa, ON, Canada
hzhu@sce.carleton.ca

Changcheng Huang
Systems and Computer Engineering
Carleton University
Ottawa, ON, Canada
huang@sce.carleton.ca

Jiayu Zhou
Computer Science and Engineering
Michigan State University
East Lansing, MI, USA
jiayuz@msu.edu

***Abstract*—The state-of-the-art mobile edge applications are generating intense traffic and posing rigorous latency requirements to service providers. While resource sharing across multiple service providers today requires a centralized, trusted repository maintained by all parties for service providers to share status. We propose EdgeChain, a blockchain-based architecture to make mobile edge application placement decisions for multiple service providers, based on a stochastic programming problem minimizing the placement cost for mobile edge application placement scenarios. All placement transactions are stored on the blockchain and are traceable by every mobile edge service provider and application vendor who consumes resources at the mobile edge.**

***Index Terms*—Mobile edge computing, blockchain, placement.**

I. Introduction

The rapid advance of mobile edge computing (MEC) has been the last mile of enabling a shared, low-latency computational environment for multi-vendor mobile edge applications. Applications with low latency tolerance, such as augmented reality (AR), video streaming, and online gaming, can deploy their services on the edge hosts at a cost, to achieve lower latency and better user experience [1]. As the market gets mature, multiple 5G service providers (SPs) can collaborate with each other in several ways for better utilization of the resources at the edge: virtual SPs have to place mobile edge (ME) applications on one of the rented edge hosts, preferably with lower cost, regardless of SPs. On the other hand, MEC base stations from different SPs can share resources with each other to process bursting requests.

For encouraging SPs to enroll their eligible MEC base stations and hosts in resource sharing, it is common to give incentives to SPs for contributing their resources of the hosts for hosting edge applications. The edge application provider will save costs and provide high-quality service with low latency to the end users. The edge host will collect incentives for its resources effectively used. The edge computing framework needs a placement service to dynamically determine the placement or removal of edge applications. The collaboration of multiple SPs and mobile edge applications vendors are posing new challenges for ME application placement from

978-1-5386-4633-5/18/$31.00 ©2018 IEEE

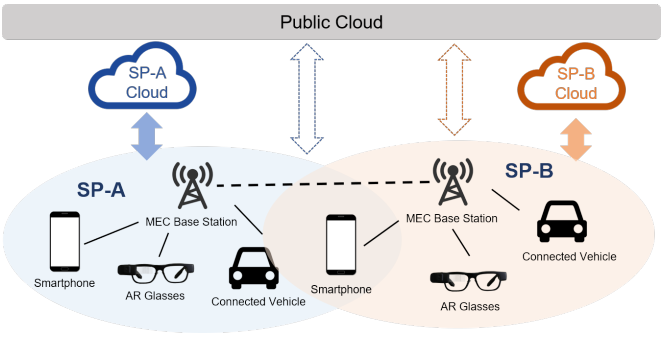

Fig. 1. **A MEC scenario in a certain service area. There are 2 ME base stations from 2 different SPs: SP-A and SP-B. They serve their own users within the service area. For resource sharing and optimization purposes, the base stations are also connected to each other.**

the following aspects: 1) A placement model has to make transparent and consistent selections of the best host for each request for edge computing resources. Moreover, the model has to take into consideration that a mobile edge application may require multiple services chained together at the edge. 2) A trusted party is required to determine the best place for application deployment. 3) The application placement service needs to be steadily available. These challenges above urge a comprehensive solution uniting all SPs and their edge hosts without bias.

In this paper, we present an architecture combined with its algorithm, namely EdgeChain, to create a decentralized placement service for mobile edge application that does not require trust to any party, i.e., trustless placement service. Compared with current placement solutions, EdgeChain has the following contributions: A cost model is presented factoring in the pricing of edge hosts, latency, and service chaining. A heuristic placement algorithm is developed based on the proposed cost model with the consideration of efficiency for running by the blockchain. We also introduce blockchain technologies to the MEC resource orchestration framework with the considerations global resource availability, allocation, and consumption information.

We divide the contents into the following sections. The related work is illustrated in Section II. Section III formulates

TABLE I
PARTIES INVOLVED IN A MEC PLACEMENT SCENARIO

Party	Description
Users	Subscribers over 5G networks with MEC enabled.
MECSPs	MEC service providers hosting *MEApps* at the network edge, close to end users.
MEAVs	Mobile edge application vendors, who provide *MEApps* and services to end users.
MEApps	Mobile edge applications provided by MEAVs.
MEHosts	Servers that belong to different MECSPs to provide hosting service of *MEApps*.
HostLinks	Network links between hosts, regardless of which MECSP they belong to.
AppLinks	Virtual links established for data transmissions traveling through the chain.

the problem. Section IV proposes the heuristic EdgeChain placement algorithm based on the problem formulation. Then the simulation results are shown in Section VI. Section VII concludes the paper.

II. Related Work

The research directions in network service chaining (NSC) were discussed in [2]. For security considerations, the authors highlighted the difficulty of bringing short-lived network services to targeted users in a single subscriber network by using the current security schemes. The potential security problems in SFC were stated in RFC7498 [3], including service overlay security, trusted classification policy, and secure SFC encapsulation. A placement problem was proposed in MEC environment with the consideration of application availability in [4]. Xiong et al. proposed a pricing strategy for offloading the blockchain's resource-consuming proof-of-work tasks to edge computing nodes [5]. A hierarchical distributed control system was built using Hyperledger Fabric blockchain [6]. The hosting locations of cloud and fog of blockchain were compared in [7] for IoT networks with the conclusion that fog nodes were better as network latency was the dominant factor. Nakamoto introduced the concept of blockchain and implemented Bitcoin [8], a decentralized cryptocurrency that first resolved the double spending problem. Blockchains are based on Merkle trees [9] to efficiently allow multiple documents to be saved together in a block. As a decentralized public ledger, blockchains can serve beyond cryptocurrencies. Ethereum [10] used blockchain to store smart contracts that support building virtually any decentralized application.

III. Problem Formulation

We first list all parties involved in a MEC placement scenario in Table I. The problem is formulated from a *MEAV*'s point of view: *MEApps* are direct consumers of the computing resources in the MEC environment, because a *MEAV* needs to pay *MECSPs* for hosting its applications in order to serve their *users* and meet the latency requirement. Each *MEApp* is equivalent to a virtual machine (VM) deployed on a *MEHost*. *MEApps* provided by different *MEAV* can be combined as a service chain to provide comprehensive services. A service chain may span multiple *MEAVs*. In this case, revenues generated by the service chain can be distributed according to the usage of each *MEApp* on the service chain. Define a chained service s as a forwarding graph [11] $G_s = (\mathbb{V}_s, \mathbb{L}_s)$, where $\mathbb{V}_s$ is the set of all *MEApps* contributing to the service, and $\mathbb{L}_s$ is the set of all *AppLinks* connecting applications together. A *MEApp* is denoted by $v \in \mathbb{V}_s$, and an *AppLink* between two *MEApps* is denoted by $l \in \mathbb{L}_s$.

The chained service is deployed on a graph of connected *MEHosts* $G_h = (\mathbb{H}, \mathbb{E})$, where $\mathbb{H}$ is the set of all *MEHosts* owned by various *MECSPs* and $\mathbb{E}$ is the set of all *HostLinks*. A *MEHost* is denoted by $h \in \mathbb{H}$, and a *HostLink* between two *MEHosts* is denoted by $e \in \mathbb{E}$. The *HostLinks* can be either physical or virtual links with fixed capacities and latencies.

Suppose in a certain service area, there are n_s *users* from various *MECSPs* requesting the same chained service s from a *MEAV*. We use m to denote a *MECSP* and h_m for a *MEHost* that belongs to m. Define an assigning function x_{vh_m}, whose value is 1 if VM v is assigned to Host h_m, and 0 otherwise.

$$x_{vh_m} \triangleq \begin{cases} 1, & v \text{ is deployed on } h_m; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Define a binary indicator of an *AppLink* between two chained *MEApps* in s, denoted by $L(v_{h_i}, v_{h_j})$, such that

$$L(v, v') \triangleq \begin{cases} 1, & l \in \mathbb{L}_s \text{ exists between } v \text{ and } v'; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Also, we use e_{ij} to represent the *HostLink* between h_i and h_j. The cost of deploying s is the sum of the cost of deploying each *MEApp* v of the service and the cost of the traffic between each two adjacent *MEApps* in the service chain shown by

$$\begin{aligned} c_s = &\sum_{h_m \in H} \sum_{v \in \mathbb{V}_s} c_{vh_m} x_{vh_m} \\ &+ \sum_{h_i, h_j \in H} \sum_{v, v' \in \mathbb{V}_s} c_{vh_i, v'h_j} x_{vh_i} x_{v'h_j} L(v, v'), \end{aligned} \tag{3}$$

where c_s represents the cost of deploying s and c_{vh_m} is for the cost of a *MEApp* v deployed on a *MEHost* h_m. We assume that the pricing scheme for the same *MECSP* is the same across all of its hosts. For a *MEHost* h_m, define its basic unit resource price, which is the unit price of serving its own subscribers, as γ_m. When h_m is serving users of other *MECSPs*, it charges a premium of δ_m for its unit resource, as the return for doing courtesy for its partners. Therefore, the shared unit resource price of h_m can be represented by $(\gamma_m + \delta_m)$. Define C_{h_m} and M_{h_m} to be the capacity of vCPU and memory provided by h_m. Define C_v and M_v as the vCPU and memory consumed by v. Define P_m to be the random variable for percentage of the *users* using the service chain s via networks of the *MECSP* m. Depending on the numbers of active users for each *MECSP*, the total cost for the *MEAV* to place its *MEApp* v onto a host of m is the cost incurred by *users* of m plus the cost by *users* of other *MECSPs*:

$$\begin{aligned} c_{vh_m} =& n_s(C_v + M_v) P_m \gamma_m \\ &+ n_s(C_v + M_v)(1 - P_m)(\gamma_m + \delta_m) \\ =& n_s(C_v + M_v) \left[P_m \gamma_m + (1 - P_m)(\gamma_m + \delta_m) \right] \\ =& n_s(C_v + M_v) \left[\gamma_m + (1 - P_m)\delta_m \right]. \end{aligned} \tag{4}$$

When a request from a *user* for a service chain arrives, the blockchain would know the *MECSP* from which the *user* subscribes. For the same placement decision, the value c_s can significantly differ over changing distribution of *users*.

An example can be two *MECSPs* m_1 and m_2, each with one host h_{m_1} and h_{m_2}. If all *users* are subscribers of m_1 and all *MEApps* are placed on *MEHosts* of m_1, then the cost payable by the *MEAV*s would be lower than if all *users* were subscribers of m_2.

A. HostLink Unit Price

The link unit price of a *HostLink* e_{ij}, denoted by $\zeta_{e_{ij}}$, is defined to describe how much to use the HostLink e_{ij}. The following two parameters will determine $\zeta_{e_{ij}}$. One is $L(v_{h_i}, v_{h_j})$ as defined in Eqn. (2). The more *AppLinks* a *HostLink* carries, the more vital and expensive it becomes. The reason behind this ranking parameter is the potential consequence of migration: failure of a *HostLink* used by many VMs would lead to massive migration of all *MEApps* connected by that *HostLink*, which would be more disruptive to the service chain. The other parameter $B_V(e_{ij})$ is the total bandwidth consumed by traffic between *MEApps* on the two hosts. It is selected because larger bandwidth usages would cause challenges at the time of migration: it can be hard to find another link with enough capacity.

$$B_V(e_{ij}) \triangleq \left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} B(v_{h_i}, v_{h_j}) \right]. \tag{5}$$

We then define the unit price $\zeta_{e_{ij}}$ of a *HostLink* e_{ij}, as the factor of the number of *AppLinks* between two hosts times the factor of traffic flowing through these links:

$$\zeta_{e_{ij}} = \frac{\left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} L(v_{h_i}, v_{h_j})\right]}{N_{e_{ij}}} \frac{B_V(e_{ij})}{B(e_{ij})}, \tag{6}$$

where $N_{e_{ij}}$ is the maximum number of virtual links possible on e_{ij}. Therefore, $\zeta_{e_{ij}} \in [0, 1]$. The value of $\zeta_{e_{ij}}$ will rise to mark up a link's importance given it is either occupied by more pairs of VMs, or there is more traffic assigned to e_{ij}, or both. The cost of any two *MEApps* is then the sum of the cost serving users that belong to the *MECSPs* owning h_i and h_j and the cost serving other users timed by the price factor $\kappa_{e_{ij}}$:

$$\begin{aligned} c_{vh_i, v'h_j} =& n_s \zeta_{e_{ij}} (P_{m_{h_i}} + P_{m_{h_j}}) \kappa_{m_{h_i} m_{h_j}} \\ &+ n_s \zeta_{e_{ij}} (1 - P_{m_{h_i}} - P_{m_{h_j}}) (\kappa_{m_{h_i} m_{h_j}} + \sigma_{m_{h_i} m_{h_j}}) \\ =& n_s \zeta_{e_{ij}} [(P_{m_{h_i}} + P_{m_{h_j}}) \kappa_{m_{h_i} m_{h_j}} \\ &+ (1 - P_{m_{h_i}} - P_{m_{h_j}}) (\kappa_{m_{h_i} m_{h_j}} + \sigma_{m_{h_i} m_{h_j}})]. \end{aligned} \tag{7}$$

B. HostLink latency

Define the latency of the link e_{ij} to be $t_{e_{ij}}$. For a service chain s, the total latency t_s is then

$$t_s = \sum_{h_i, h_j \in \mathbb{H}} \sum_{v_{h_i}, v_{h_j} \in \mathbb{V}_s} L(v_{h_i}, v_{h_j}) x_{vh_i} x_{vh_j} t_{e_{ij}}. \tag{8}$$

In the equation above, $t_{e_{ij}}$ is a constant depending on the particular e_{ij}. If $h_i = h_j$, then we consider the latency to be 0, since no actual *HostLink* is used for data transmission between the two *MEApps*. Define the maximum latency allowed for the service chain s is T_s. Then there must be $t_s \leq T_s$ to meet the latency requirement.

C. Stochastic Programming Formulation

The problem is formulated as a stochastic programming optimization. Define $\mathbb{V}_h$ as the set of all *MEApps* deployed on the *MEHost* h. The objective is to minimize the total cost of the service chain s to provide service with the lowest cost to the end user:

Minimize

$$\begin{aligned} c_s =& \sum_{h_m \in \mathbb{H}} \sum_{v \in \mathbb{V}_s} c_{vh_m} x_{vh_m} \\ &+ \sum_{h_i, h_j \in \mathbb{H}} \sum_{v, v' \in \mathbb{V}_s} c_{vh_i, v'h_j} x_{vh_i} x_{v'h_j} L(v, v') \\ =& \sum_{h_m \in \mathbb{H}} \sum_{v \in \mathbb{V}_s} x_{vh_m} n_s (C_v + M_v) \left[\gamma_m + (1 - P_m)\delta_m\right] \\ &+ \sum_{h_i, h_j \in \mathbb{H}} \sum_{v, v' \in \mathbb{V}_s} n_s \zeta_{e_{ij}} [(P_{m_{h_i}} + P_{m_{h_j}}) \kappa_{m_{h_i} m_{h_j}} \\ &+ (1 - P_{m_{h_i}} - P_{m_{h_j}}) (\kappa_{m_{h_i} m_{h_j}} + \sigma_{m_{h_i} m_{h_j}})] L(v, v'), \end{aligned} \tag{9}$$

$$w.r.t. \quad x_{vh_m},$$

$$s.t. \quad B(e_{ij}) \geq \sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} B(v_{h_i}, v_{h_j}), \tag{10}$$

$$C_h \geq \sum_{v \in \mathbb{V}_h} C_v, \tag{11}$$

$$M_h \geq \sum_{v \in \mathbb{V}_h} M_v, \tag{12}$$

$$\sum_{h_i, h_j \in \mathbb{H}} \sum_{v_{h_i}, v_{h_j} \in \mathbb{V}_s} L(v_{h_i}, v_{h_j}) x_{vh_i} x_{vh_j} t_{e_{ij}} \leq T_s. \tag{13}$$

IV. The EdgeChain Placement Algorithm

The formulation presented in the previous section is a stochastic programming problem. Problems of this type been proved to be NP-hard [12]. It may not be computationally feasible when attempting to solve it in large scale. To apply our model to real-world scenarios, we design a heuristic algorithm called EdgeChain to achieve suboptimal results by applying a hybrid strategy of best-fit and first-fit decreasing algorithm. The pseudo code of the algorithm is shown in Algorithm 1.

V. EdgeChain Design and Implementation

The implementation of EdgeChain takes requests to place *MEApps* from *MEAVs*, and the placement algorithm runs as the smart contract on a capable blockchain, namely VeChain Thor [13], to select the best *MEHost* from all candidates. The NFV orchestrator of the related *MECSP* receives and enforces the placement decision, while posting the transaction onto the blockchain for recording.

A. EdgeChain Work Flow

A typical EdgeChain work flow can be demonstrated by Fig. 2, where there are three parties participating in the entire process: *MECSPs*, *MEAVs*, and mining nodes. We use circled numbers and alphabets to define the work flow in sequence.

Algorithm 1: EdgeChain Placement Algorithm

Data: host_list: list of candidate MEHosts
Data: app: requested *MEApp* to be placed, including its max latency allowed, stored in latency
Data: max_latency: max latency allowed for the service chain
Result: The best *MEHost* in host_list to place app, or none if no valid host is found

```
begin
    sort by percentage of users of the service chain descending
    if multiple MEHosts found then
        sort host_list by the locations of app's last-hop MEApps
        if still multiple MEHosts found then
            sort by the latency of the HostLinks to the previous MEApps in the service chain ascending
        end
    end
    for h ∈ host_list do
        latency ← all latencies added together if app placed on h
        if latency ≤ max_latency then
            cpu_left ← calculate remaining vCPU by C_h and C_v of each MEApp placed on h
            mem_left ← calculate remaining memory by M_h and M_v of each MEApp placed on h
            if cpu_left ≥ 0 and mem_left ≥ 0 then
                return h
            end
        end
    end
    return none
end
```

(1) A *user* requests a service chain from the blockchain. (2) The request for the service chain is recorded. Then the requests for *MEApps* are propagated to all corresponding *MEAVs*. (3) Based on its user demand, the *MEApp* Scheduler decides to create a new instance of MEApp and pass the request to the blockchain client of the *MEAV*. (4) The blockchain client running the EdgeChain service sends the request to the blockchain, creating records for the request of placing a new *MEApp*. (5) The request of creating a new *MEApp* arrives at a MECSP through its blockchain client. (6) For every *MECSP*, the blockchain client requests the NFV Orchestrator (NFVO) to call the EdgeChain placement algorithm downloaded to the resource manager for the decision of the placement. This will ensure that the placement algorithm be executed by different parties for verifying the results. (7) The NFVO calls the EdgeChain placement algorithm for the placement decision. (8) NFVO sends the request to place the *MEApp* to the VNF Manager (VNFM). Also, a transaction shown in Fig. 3 will be posted to the blockchain to record that placement actually occurs. (9) The VNFM sends the request to the NFV Infrastructure (NFVI) deploy the *MEAPP* onto the target *MEHost*. (T) The mining nodes periodically perform the mining process to verify the blockchain, as well as earning tokens for requesting placement services.

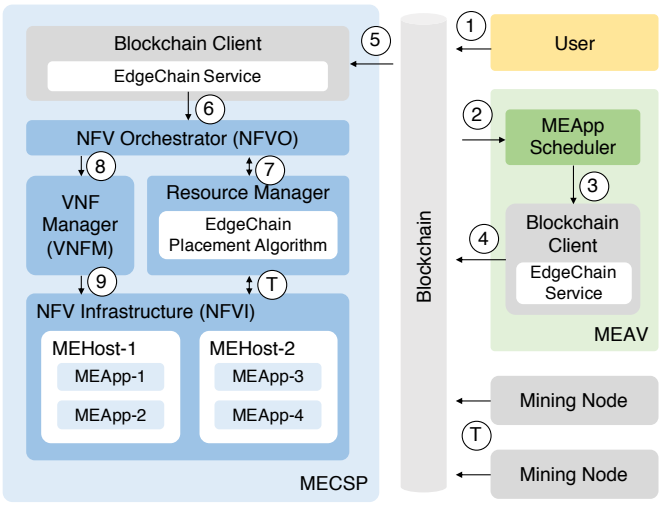

Fig. 2. **Typical work flow of EdgeChain. *MECSPs*, *MEAVs*, and mining nodes participate in the process. Steps of the work flow are marked by circled numbers and alphabets with details documented in Section V-A.**

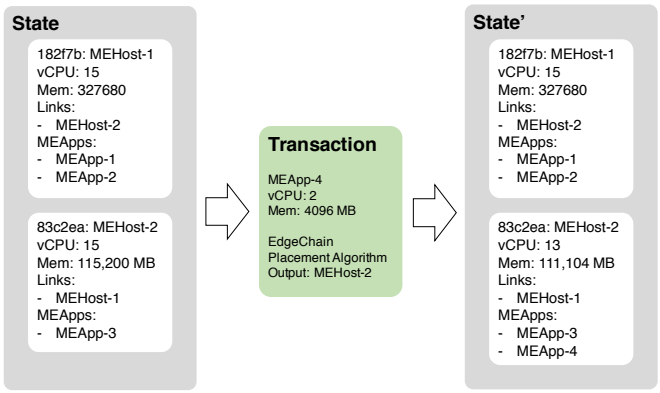

Fig. 3. **A placement transaction in EdgeChain. A state transition happens upon a transaction. As this figure shows, *MEApp-4* is to be placed with the requirement of 2 vCPUs and 4096 MB of memory. The input of the EdgeChain placement algorithm is the current state of the two *MEHosts*. The result is to place *MEApp-4* onto *MEHost-2*. After the transaction is accepted, the resources taken by *MEApp-4* are deducted from the remaining resources of *MEHost-2*.**

VI. Numerical Results

The following assumptions are made to simplify the modeling of the problem without losing generality. We first discuss the placement results output by the EdgeChain algorithm for the same service chain on the same set of *MEHosts*. 1) The unit costs of the CPU and memory of all hosts for the same *MECSP* are the same. 2) Costs of network bandwidth for all links follow the same unit price. 3) One mobile edge application includes the same type of VMs with the same CPU, memory and network bandwidth requirements. 4) A request from the user will be processed by one VM, while the VM may communicate with other VMs to exchange information.

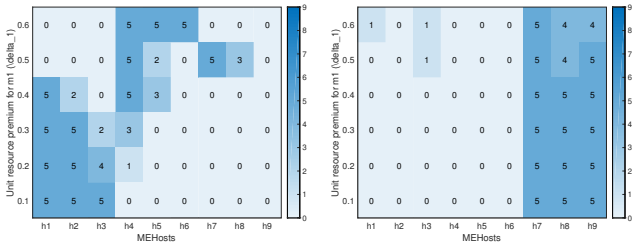

Fig. 4. **Placement results of 3 service chains consisting of 15 *MEApps* in all. δ_{m_1} changes from 0.1 to 0.6. The left table shows the placement decision when $P_{m_1} = 0.5$ and $P_{m_2} = P_{m_3} = 0.25$. The right table shows the placement decision when $P_{m_1} = P_{m_1} = 0.25$ and $P_{m_3} = 0.5$.**

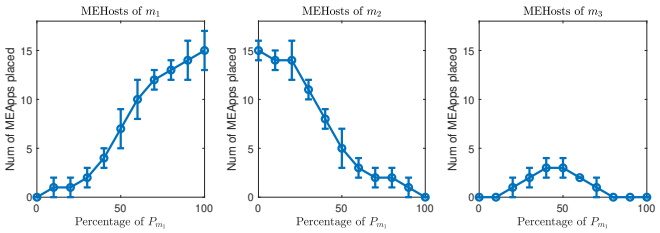

Fig. 5. **Numbers of *MEApps* placed on the 3 *MEHost* with different percentages of *users* in the network. Users of m_1 increase from 0% to 100%, while those of m_2 decrease from 100% to 0%. There is no user for m_3.**

A. Parameters

First, we choose a MEC service scenario of 3 *MECSPs* m_1, m_2, and m_3, each with 3 *MEHosts*, where h_1, h_2, h_3 belong to m_1, h_4, h_5, h_6 belong to m_2, and h_7, h_8, h_9 belong to m_3. Three identical requested service chain, each with 5 *MEApps* is to be placed. The *MEApps* of each service chain are denoted by v_1, v_2, v_3, v_4, and v_5. The service chain starts from v_1 and ends at v_5: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$. We assume that all *MEApps* have the same CPU, memory and bandwidth requirements shown in Table II, along with other parameters.

TABLE II
PARAMETERS FOR THE MEC SCENARIO

Parameter	Value	Parameter	Value
C_v	2 vCPUs	M_v	2048 MB
C_h	64 vCPUs	M_h	65536 MB
γ_{m_1}	1.0	δ_{m_1}	0.2
κ_{m_1}	1.0	σ_{m_1}	0.2
γ_{m_2}	0.8	δ_{m_2}	0.5
κ_{m_2}	0.8	σ_{m_2}	0.5
γ_{m_3}	1.2	δ_{m_3}	0.3
κ_{m_3}	1.2	σ_{m_3}	0.3
n_s	100 users	P_m	*var*
$B(e_{ij})$	10000 Mbps	$B(v, v')$	30 Mbps
$t_{e_{ij}}$	15 ms	T_s	50 ms

B. Placement trends with changing unit resource premium

The placement decision changes by the increase of δ_{m_1} under different user distributions are shown in Fig. 4, where δ_{m_1}, the unit resource premium payable to the *MECSP* for hosting *MEApps* for others, increases from 0.1 to 0.6. From the results of the two scenarios, we learn that the *MEHosts* with lower combination of unit resource base price (γ_m) and unit resource premiums (δ_m) will be selected first. The *MEHosts* of the *MECSP* will have more weight upon consideration if there are more *users* from that *MECSP*.

C. Placement trends with changing user distribution

We simulate various scenarios with different percentages of *users* for m_1 and m_2, while there is no user for m_3. Users of m_1 increase from 0% to 100%, while those of m_2 decrease from 100% to 0%. The results show the trends of *MEApps* migrating to *MEHosts* owned by the *MECSP* that has more active users to avoid premiums charged by other *MECSPs*. However, resource sharing still takes place (m_3 hosting *MEApps* for m_1 and m_2) when needed for better latency results and service quality.

VII. CONCLUSIONS

In this paper, we have presented the architecture and the algorithms for mobile edge applications placement for multiple mobile edge computing service providers, leveraging the blockchain-based system called EdgeChain. Future work will be considering multiple service chains initiated by multiple users, to achieve lower overall costs for the entire system.

VIII. ACKNOWLEDGMENT

We thank VeChain Foundation for providing support in algorithm integration with VeChain Thor.

REFERENCES

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI White Paper*, vol. 11, 2015.
[2] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov 2013, pp. 1–7.
[3] P. Quinn and T. Nadeau, "Problem statement for service function chaining," 2015.
[4] H. Zhu and C. Huang, "Availability-aware mobile edge application placement in 5g networks," in *IEEE Globecom'17*, Dec 2017.
[5] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Edge computing resource management and pricing for mobile blockchain," *arXiv preprint arXiv:1710.01567*, 2017.
[6] A. Stanciu, "Blockchain based distributed control system for edge computing," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, May 2017, pp. 667–671.
[7] M. Samaniego and R. Deters, "Blockchain as a service for iot," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Dec 2016, pp. 433–436.
[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[9] D. Bayer, S. Haber, and W. S. Stornetta, *Improving the Efficiency and Reliability of Digital Time-Stamping*. New York, NY: Springer New York, 1993, pp. 329–334.
[10] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.
[11] G. Brown, "Service chaining in carrier networks," 2015.
[12] A. A. Gaivoronski, A. Lisser, R. Lopez, and H. Xu, "Knapsack problem with probability constraints," *Journal of Global Optimization*, vol. 49, no. 3, pp. 397–413, 2011.
[13] VeChain Foundation, https://www.vechain.org/, 2018, [Online; accessed 10-Jan-2018].