Received: 00 Month 0000 | Revised: 00 Month 0000 | Accepted: 00 Month 0000

DOI: xxx/xxxx

RESEARCH ARTICLE

# EdgePlace: Availability-aware Placement For Chained Mobile Edge Applications

He Zhu* | Changcheng Huang

[1]Department of Systems and Computer Engineering, Carleton University, ON, Canada

**Correspondence**
*He Zhu, Email: hzhu@sce.carleton.ca

**Present Address**
Department of Systems and Computer Engineering, Carleton University

**Summary**

Mobile edge computing (MEC) literally pushes cloud computing from remote datacenters to the life radius of end users. By leveraging the widely adopted ETSI network function virtualization (NFV) architecture, MEC provisions elastic and resilient mobile edge applications with proximity. Typical MEC virtualization infrastructure allows configurable placement policy to deploy mobile edge applications as virtual machines (VMs): affinity can be used to put VMs on the same host for inter-VM networking performance, while anti-affinity is to separate VMs for high availability. In this paper, we propose a novel model to track the availability and cost impact from placement policy changes of the mobile edge applications. We formulate our model as a stochastic programming problem. To minimize complexity challenge, we also propose heuristic algorithm called EdgePlace. With our model, the unit resource cost increases when there are less resources left on a host. Applying affinity would take up more resources of the host but saves network bandwidth cost because of co-location. When enforcing anti-affinity, experimental results show increases of both availability and inter-host network bandwidth cost. For applications with different resource requirements, our model is able to find their sweet points with the consideration of both resource cost and application availability, which is vital in a less robust MEC environment.

**KEYWORDS:**
Mobile Edge Computing, 5G, Placement Policy, Stochastic Optimization, Cloud Computing

## 1 | INTRODUCTION

Mobile edge computing (MEC) is taking Network Function Virtualization (NFV) to the end users closer than ever[1,2,3]. Instances of edge computing, including regional datacenters[4], cloudlets[5], and fog nodes[6], deliver highly-responsive cloud services at the network edge. As key technologies towards 5G, MEC architecture proposed by ETSI[3] leverages existing NFV frameworks widely adopted by carriers and vendors[7,8]. With the focus on IoT devices, OpenFog[9] is another MEC framework which targets extending elements of compute, networking and storage across the cloud through to the edge of the network. Elastic mobile edge applications, including network services, are deployed close to the user equipment (UE), which is any device used directly by an end-user for communication, with low latency. Both UE application providers and telecommunication service providers (TSPs) can take advantage of MEC to reduce cost and to adjust services with agility based on fast-changing user demands.

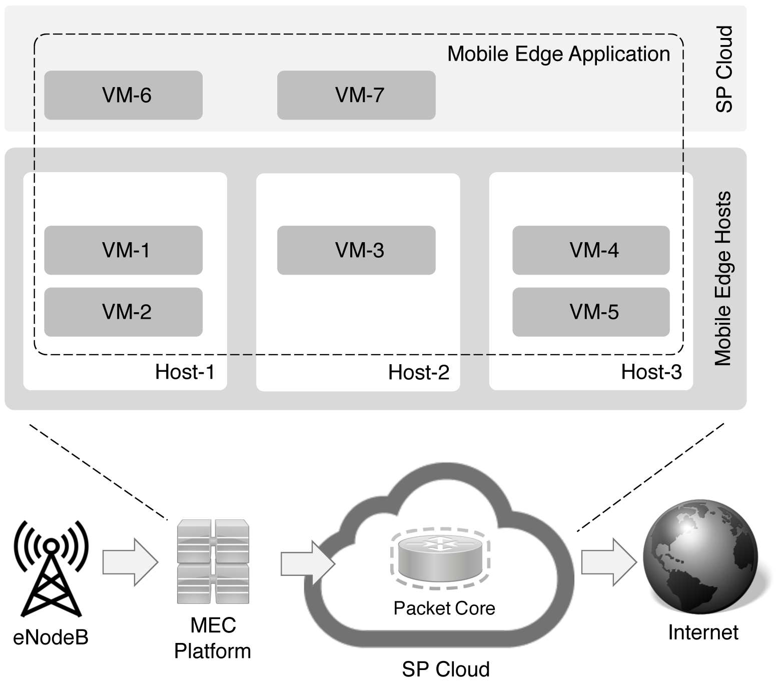

**FIGURE 1 A mobile edge application deployment with host placement rules. There are five VMs deployed in three groups with each group placed on a separate host. A minimum of three VMs are required for the application. The placement will ensure the application is in service if one host is down.**

A mobile edge application consists of one or more collaborating virtual machines (VMs). It is of paramount importance to maintain the high availability of mobile edge applications. Compared to centralized datacenters used by public cloud, MEC hosts are heterogeneous with varying computing, storage and networking capabilities[10]. Smaller scale private cloud servers can be deployed near their designated groups of users as MEC hosts, the characteristics of which lead to the following indications:

(i) A single MEC server deployment is less powerful compared to the highly-available, centralized cloud as it serves a smaller group of users within the base station's coverage. It can be a micro datacenter which is unlikely to merit its own security guard or have the same level of redundancy as a larger facility[11].

(ii) The offloading nature of MEC brings higher system complexity that can jeopardize the availability[12].

(iii) Service function chaining (SFC) is possible on MEC servers, as videos, augmented reality data, location-based services, and other computational-intensive tasks can take a chain of services to process.

These facts conclude that the hosts used in MEC are less reliable with lower availability. When mobile edge applications run in MEC servers, they must be protected from service outage due to host failure.

To maximize the availability while maintaining costs and latencies at acceptable levels, placement rules often come into play to tune the performance and security of a mobile edge application[13]. In practice, placement rules mainly refer to the affinity and anti-affinity rule[14]. A group of VMs with the affinity rule applied must be deployed on the same host. On the contrary, the anti-affinity rule to a group of VMs ensures that each VM in the group is deployed on a different host.

The affinity rule helps reduce communication costs between VMs serving the same mobile edge application: VMs on the same host connect to each other using virtual networks private to the host and require no physical networking infrastructure. Same-host network traffic essentially takes up computational resources of the host and has better performance than physical networks. This becomes handy especially when frequent inter-VM communications are needed. An obvious down side of the affinity rule is putting all eggs in one basket. If the host is down, the entire mobile edge application would be out of service. Resource contention is another drawback due to oversubscription, which is typically configured for maximizing host resource utilization[15]. Too many resource-thirsty VMs packed together would overload their host.

In comparison, anti-affinity rules are ideal for High Availability (HA). If multiple VMs of the same type are deployed on different hosts, having one host down would not take all instances out of service. Therefore, the mobile edge application can still be functional. Fig. 1 demonstrates an example of a mobile edge application deployment across multiple hosts to increase availability. On the other hand, enforcing the anti-affinity rule for resource-intensive VMs can reduce chances of the hosts being overloaded due to oversubscription. As a trade-off, the mobile edge application using anti-affinity rule would lose the benefits of low communication costs and higher confidentiality brought by co-location.

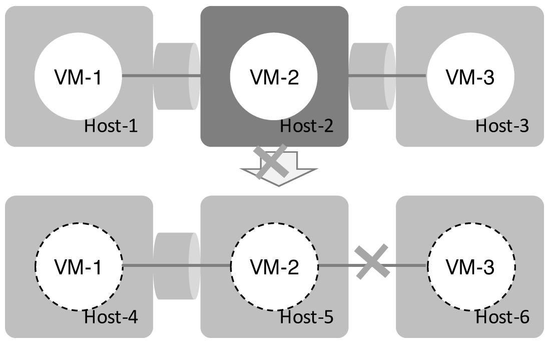

**FIGURE 2 An attempt to migrate a service chain consisting of 3 VMs and two links using L2 switching. If Host-2 is down, the service chain has to be migrated to three hosts with the same topology. Host-4, Host-5, and Host-6 have enough resources to have the service chain deployed. However, the topology is not identical. Therefore, the attempt to migrate the service chain will not succeed.**

Besides hosts availability, Service Function Chaining (SFC)[16] is commonly adopted to formulate a network function with complete features to provide end-to-end service. Similarly, SFC can be required for a mobile edge service provisioned by multiple chained functions. Different functions, i.e., VMs deployed on hosts, must be chained together to process a stream of requests. Therefore, network topology and its availability can become the primary bottleneck of the mobile edge service. When a link is down, functions connected by that link will need to migrate to recover from the link outage. For instance, when the SFC is formulated by L2 switching, the open vSwitch (OVS) will be responsible for switching traffic among the functions, which means all applications can only be deployed on the same host. The outage of the virtual switch would lead to the migration of all functions to a different host. Therefore, it is more difficult to migrate a mobile edge application considering SFC. Fig. 2 shows an example of attempt to migrate a service chain due to Host 2's outage. Host 4, 5 and 6 are almost a valid combination to host the service, except there is no link between Host 5 and 6. As a result, the migration attempt would fail.

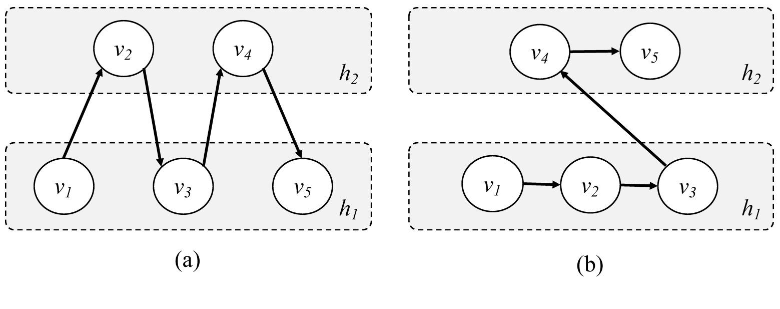

**FIGURE 3 Inefficient partitioning of VMs for a mobile edge application can cause unnecessary inter-host traffic. In (a), SFC traffic flow would travel between the two hosts 4 times, while it would travel only once in (b).**

Enforcing SFC can also cause massive increase of latency if the placement strategy is not aware of the chaining policy. As demonstrated in Fig. 3 (a), inefficient partitions of VMs with SFC applied can cause traffic going back and forth among hosts, resulting in excessive latency in comparison with Fig. 3 (b).

Considering placement constraints of host availability, SFC, and the limited resources at the mobile edge, it can be foreseen that when mobile edge hosts are unavailable, mobile edge application VMs might not be able to continue service if they are kept

**TABLE 1** Notations Used in Problem Formulation

| Notation | Description |
|---|---|
| $\mathbb{V}, N_V, v, \mathbb{H}, N_H, h, c$ | $\mathbb{V}$ is the set of $N_V$ VMs provisioning a VNF, each VM denoted by $v$. $\mathbb{H}$ is the set of $N_H$ hosts available for VNF deployment, each host denoted by $h$. $c$ stands for the remote cloud location. |
| $e_{ij}, a_{ij}$ | $e_{ij}$ is the network link between the two hosts $h_i$ and $h_j$. $a_{ij}$ is the maximum number of virtual links possible on $e_{ij}$. |
| $x_{vh}, x_{vc}$ | If $v$ is deployed on $h$, then $x_{vh} = 1$. Otherwise, $x_{vh} = 0$. If $v$ is deployed on the cloud, then $x_{vc} = 1$. Otherwise, $x_{vc} = 0$. |
| $N_m$ | Minimum number of active VMs required by the VNF. |
| $P_V, P_H, P_E(e_{ij})$ | $P_V$ is the probability of a VM working without internal failure. $P_H$ is the probability of a host not failing. $P_E(e_{ij})$ is the probability that $e_{ij}$ is up and available |
| $\hat{v}_h, \hat{v}_c$ | Total number of VMs assigned to host $h$ and the cloud $c$. |
| $A_h, p_{a_h}$ | $A_h$ is the random variable of the number of VMs available on $h$. $p_{a_h}$ is the probability there are $a_h$ VMs available on $h$. |
| $p_{\bar{a}}, p_E(\bar{a}), p_y$ | $p_{\bar{a}}$ is the probability there are $(a_1, a_2, ..., a_H)$ VMs on hosts $(h_1, h_2, ..., N_H)$, given $\bar{a} = (a_1, a_2, ..., a_H)$. $p_E(\bar{a})$ is the probability all hosts with one or more VMs deployed are connected to each other. $p_y$ is the probability at least $y$ VMs available in total. |
| $B(v_{h_i}, v_{h_j})$ | Total bandwidth demand from $v_{h_i}$ to $v_{h_j}$. |
| $B(e_{ij}), R_B(e_{ij})$ | Total and remaining bandwidth of link $e_{ij}$. |
| $C_h, M_h, B_c$ | $C_h$ and $M_h$ are the total capacities of vCPU and memory of host $h$. $B_c$ is the total bandwidth between the mobile edge and the cloud. |
| $C(v), M(v)$ | Number of vCPUs and amount of memory required to deploy $v$. |
| $C_v, M_v$ | Number of vCPUs and amount of memory required to complete $v$'s own tasks, excluding resource consumed to coordinate with other VMs. |
| $\gamma_C, \gamma_M$ | Proportional ratio between the number of vCPUs/amount of memory required to coordinate with other VMs and the number of VMsto communicate. |
| $\alpha_C, \alpha_M$ | Conversion ratio from the intra-host unit network bandwidth usage to the unit CPU/ memory usage. |
| $R_C(h/c), R_M(h/c)$ | Remaining number of vCPUs and remaining amount of memory on host $h$/cloud. |
| $S_C(h/c), S_M(h/c)$ | Unit cost of consuming vCPUs and memory resources of $h$/cloud. |

at the edge. To maintain the desired availability, the more reliable, centralized cloud can come to the picture to coordinate with the edge.

With the host placement rules and the link availability requirements, in this paper, our interest is in finding adaptive placement strategies for different types of mobile edge applications to achieve lower costs, while still satisfying the availability and confidentiality requirements. Compared to existing work, our contributions include the following:

(i) We address the application availability concerns even when some mobile edge hosts are down. We believe that hosts tend to be less reliable at the network edge and availability issues of MEC applications need more attention.

(ii) We consider the link health between hosts to support SFC. Individual VMs will be migrated when one or more links are down to maintain the service of the SFC-enabled mobile edge application.

(iii) A cost model is built considering the factors of inter-host traffic and resource over-committing, to balance the load without causing explosive traffic between hosts.

(iv) We formulate a stochastic programming problem to minimize the cost based on our cost model, while maintaining the availability requirements.

(v) A heuristic algorithm, namely EdgePlace, is developed to return suboptimal results as the problem scales. The cost model and EdgePlace placement algorithm can fit all industrial standards to address the availability concerns in MEC environments. Numerical results show the effectiveness of EdgePlace.

We divide the contents into the three following sections. Section 2 formulates the problem. Then the experimental results are shown in Section 3. The related work is illustrated in Section 4. Section 5 concludes the paper.

## 2 | PROBLEM FORMULATION

For the ease of reference, the notations used when we formulate the problem are listed in Table 1 . Suppose a mobile edge application has a set of elastic group of VMs, denoted by $\mathbb{V}$, to be deployed on a MEC virtualization infrastructure (MECVI) with a set of hosts $\mathbb{H}$. VMs can be deployed on any of the hosts available from the MECVI, or on the remote cloud, denoted by $c$. Assume that there be $N_V$ VMs used by the mobile edge application, with each VM denoted by $v$, and $N_H$ hosts in the MECVI, with each host denoted by $h$. Different hosts are connected to each other by network links. We denote the network link between Hosts $h_i$ and $h_j$ as $e_{ij}$.

Define an assigning function $x_{vh}$, whose value is 1 if VM $v$ is assigned to Host $h$, 0 otherwise. For the cloud, a similar assigning function $x_{vc}$ is defined to be 1 if $v$ is deployed on $c$ and 0 if not.

$$\begin{aligned} x_{vh} &= \begin{cases} 1, & v \text{ is deployed on } h; \\ 0, & \text{otherwise.} \end{cases} \\ x_{vc} &= \begin{cases} 1, & v \text{ is deployed on the cloud;} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{1}$$

### 2.1 | Availability of Elastic Mobile Edge Applications

Let the minimum number of VMs required by the mobile edge application be denoted by $N_m$. Similar to virtual network function (VNF) resource management in service chaining[17], if the number of available VMs is at least $N_m$ and they are connected according to the designed topology, the mobile edge application is then considered in service. Otherwise, it is deemed down as it would not satisfy SLA requirements for the volume of requests.

An intuitive way to increase the availability of the mobile edge application is VM redundancy. In production environments, for example, it is quite common to keep a certain number of VMs of the same type running with the configuration of `Keepalived`[18], to maintain the service availability or to balance the load. Thanks to the elasticity of the mobile edge application, auto-scaling is enabled in form of deploying extra VMs, so that even if some VMs are down, there are still more than $N_m$ VMs in service. There must be $N_V \geq N_m \geq 0$. On the other hand, it is not as easy to increase link redundancy: substrate network links between hosts are pre-created and there isn't always a match for a whole application with SFC.

To keep a mobile edge application up, the availability of both the VMs and the links is required. Consider the failure points of these two factors, there are three situations in our discussion that can lead to VM service outage:

- ⋆ *Application internal failure*. If the software installed crashes or hangs, the service provided by the VM would be unavailable. Internal failure on one VM is assumed to be independent from those on other VMs. Denote the probability that a VM is working without internal failure as $P_V$.
- ⋆ *Host failure*. If one host is down, all VMs deployed on it would be out of service. Denote the probability that a host will not fail as $P_H$.
- ⋆ *Link failure*. A link may experience technical issues, either due to a software bug, or substrate network outage. If a link used by a VNF is down, it may affect the availability of the VNF.

Define $\hat{v}_h$ as the total number of VMs assigned to host $h$, and $\hat{v}_c$ as the number of VMs assigned to the cloud. Then we have

$$\hat{v}_h = \sum_v x_{vh}, \quad \hat{v}_c = \sum_v x_{vc}. \tag{2}$$

Let $A_h$ be the random variable denoting the number of VMs to be available on a host $h$. Because different VMs on the same host fail independently due to internal failure, by binomial distribution, we have

$$p_{a_h} \triangleq \Pr\left\{A_h = a_h\right\} = \binom{\hat{v}_h}{a_h} P_V^{a_h}(1 - P_V)^{\hat{v}_h - a_h} P_H. \tag{3}$$

By assumption, different hosts also fail independently. We have

$$p_{\bar{a}} \triangleq \Pr\left\{A_1 = a_1, A_2 = a_2, ..., A_H = a_H\right\} = \prod_h p_{a_h}. \tag{4}$$

The additional condition for all the VMs to be available to the mobile edge application is to ensure that the links among these VMs are all available, too. Define $p_E(\bar{a})$ as the probability that all hosts with one or more VMs deployed are connected to each other. Also, define $P_E(e_{ij})$ as the probability that $e_{ij}$ is up and available. We have

$$p_E(\bar{a}) \triangleq \prod_{a_{h_i}, a_{h_j} > 0, a_{h_i} \neq a_{h_j}} P_E(e_{ij}). \tag{5}$$

Define $p_y$ as the probability that there are at least $y$ VMs available and $\hat{a} = \sum_h a_h$. Meanwhile, all hosts with VMs deployed must be able to communicate to each other. We have

$$\begin{aligned} p_y &\triangleq \Pr\left\{\sum_h A_h \geq y - \hat{v}_c\right\} p_E(\bar{a}) \\ &= \sum_{\bar{a}, \hat{a} = y - \hat{v}_c}^{N_V} p_{\bar{a}} \prod_{a_{h_i}, a_{h_j} > 0, a_{h_i} \neq a_{h_j}} P_E(e_{ij}) \\ &\geq 1 - \eta, \end{aligned} \tag{6}$$

where $\bar{a} = (a_1, a_2, ..., a_H)$ and $\eta$ is a small positive number denoting the maximum failure probability allowed.

## 2.2 | Inter-host Link Availability and Link Importance Factor

Inter-host link availability is a fundamental part to ensure the availability of the mobile edge application. An event of a key inter-host link outage is catastrophic: even if all individual VMs are running, the traffic would not be able to flow through between one or more pairs of VMs and the SFC would not be functional. For each inter-host link $e_{ij}$, there can be one or more inter-VM links sharing its bandwidth. Link outages require migrating the mobile edge application VMs if the network links cannot be fixed in time. Therefore, link availability has significant influence on possible VM migrations and costs incurred.

The link importance factor of an inter-host link $e_{ij}$, denoted by $I_L(e_{ij})$, describes how important an inter-host link $e_{ij}$ is for the application availability. We determine the importance of each host link by the two parameters below.

The first parameter is the indicator of a link between two individual VMs, denoted by $L(v_{h_i}, v_{h_j})$, such that

$$L(v_{h_i}, v_{h_j}) = \begin{cases} 1, & \text{there is traffic between } v_{h_i} \text{ and } v_{h_j}; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The more inter-VM links an inter-host link carries, the more vital it becomes. The reason behind this ranking parameter is the potential consequence of migration: failure of an inter-host link used by many VMs would lead to massive migration of all VMs connected by that inter-host link, which would be more disruptive to the service. When placing VMs, more reliable host links should be picked if it will be intensively shared.

The other parameter is $B_V(e_{ij})$, which is the total bandwidth consumed by traffic between VMs 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.

$$B_V(e_{ij}) = \left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} B(v_{h_i}, v_{h_j})\right]. \tag{8}$$

Combining the two parameters, we define the link importance factor of a host link $e_{ij}$, denoted by $I_L(e_{ij})$, as the number of virtual links between two hosts times the traffic flowing through the link:

$$I_L(e_{ij}) = \frac{\left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} L(v_{h_i}, v_{h_j})\right]}{a_{ij}} \frac{B_V(e_{ij})}{B(e_{ij})}, \tag{9}$$

where $a_{ij}$ is the maximum number of virtual links possible on $e_{ij}$. Therefore, $I_L(e_{ij}) \in [0, 1]$. The value of $I_L(e_{ij})$ 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.

## 2.3 | Inter-host Network Bandwidth Costs

High availability comes at a cost: extra resources are used for hosting VMs, and extra traffic occurs between VMs. The traffic between VMs on the same host will be processed by the CPU of the host without going through the actual physical network. In this section, we temporarily ignore the cost of intra-host traffic. It will be discussed in the next section as part of the CPU cost.

Let $v_h$ represent a VM deployed on Host $h$, i.e., $x_{v_h h} = 1$. Consider a VM $v_{h_i}$ on Host $h_i$ sends traffic to another VM $v_{h_j}$ on Host $h_j$. We model the bandwidth demand for every two VMs in the mobile edge application, rather than simply viewing VMs on the same host as a cluster. This is due to SFC: even if two VMs are on the same host, there can be no traffic between them as they may not be next to each other in the chain. Let $B(v_{h_i}, v_{h_j})$ be the total bandwidth demand from $v_{h_i}$ to $v_{h_j}$. If there is too much inter-host traffic, the networks would become congested and fail the mobile edge application. The required traffic throughput between two hosts must not exceed the designed bandwidth for the inter-host network. We define the total bandwidth of $e_{ij}$ by $B(e_{ij})$ and its residue bandwidth by $R_B(e_{ij})$. We calculate the residue bandwidth as following:

$$R_B(e_{ij}) = B(e_{ij}) - B_V(h_i, h_j). \tag{10}$$

Let $w_B(e_{ij})$ stand for the unit cost of consuming the bandwidth of $e_{ij}$. We model $w_B(e_{ij})$ to be inversely proportional to $R_B(e_{ij})$ with the constant of proportionality $W_B$. Such model will favor choosing those links among hosts with more residual bandwidths and therefore will achieve balancing bandwidth consumption across the links between hosts. This reduces the risk of increasing delay due to link congestion.

Define the bandwidth cost of $e_{ij}$ as $S_B(e_{ij})$. Additionally, we model $S_B(e_{ij})$ to be inversely proportional to the availability of $e_{ij}$, which factors in the potential cost to migrate the SFC in event of a link failure: the more likely the link is failing, the higher price it would cost to use that link. Then we have

$$\begin{aligned} S_B(e_{ij}) &= \frac{B_V(e_{ij}) w_B(e_{ij}) I_L(e_{ij})}{P_E(e_{ij})} \\ &= \frac{B_V(e_{ij}) W_B}{R_B(e_{ij})} \frac{\left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} L(v_{h_i}, v_{h_j})\right]}{a_{ij} P_E(e_{ij})} \frac{B_V(e_{ij})}{B(e_{ij})} \\ &= \frac{B_V^2(e_{ij}) W_B \left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} L(v_{h_i}, v_{h_j})\right]}{B(e_{ij}) \left[B(e_{ij}) - B_V(e_{ij})\right] a_{ij} P_E(e_{ij}) + \delta}, \end{aligned} \tag{11}$$

where $\delta$ is a small positive number to avoid dividing by zero.

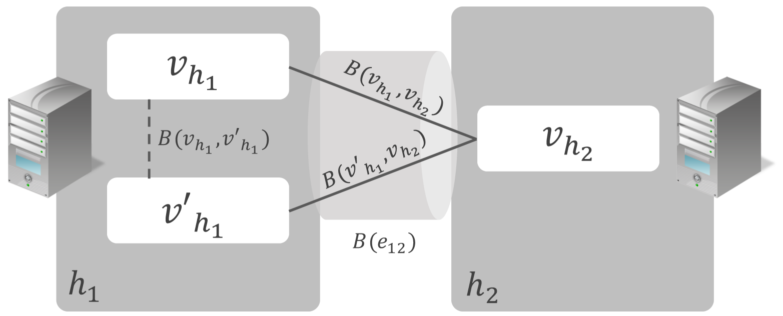

**FIGURE 4 Inter-host network bandwidth consumptions $B(v_{h_1}, v_{h_2})$ and $B(v'_{h_1}, v_{h_2})$ take up the bandwidth of $e_{12}$ and incur bandwidth costs. Intra-host network bandwidth consumption $B(v_{h_1}, v'_{h_1})$ consumes extra vCPU and memory of $h_1$ and incurs vCPU and memory costs.**

## 2.4 | CPU and Memory Costs

Consider any two VMs of the same mobile edge application. When they coordinate with each other, they will leave CPU and memory footprints on the host(s). If they are on two separate hosts $h_i$ and $h_j$, the networking costs are reflected by $S_B(e_{ij})$ as shown in last section. If they are on the same host, no bandwidth cost will incur. However, splitting workloads across VMs on the same host also consumes resources. Instead of bandwidth, it costs extra host CPU and memory resources by running on virtual networks. Fig. 4 shows examples of costs from both inter- and intra-host traffic between two VMs.

Define the number of vCPUs required by $v$ as $C(v)$. It can be divided into two parts. One fixed part is to complete its own tasks, denoted by $C_v$. The other part is to coordinate with other VMs. Let the number of vCPUs required to coordinate with other VMs be proportional to the number of VMs to communicate with constant $\gamma_C$. We have $C(v) = C_v + \gamma_C(N_V - 1)$.

Let the conversion ratio from the intra-host unit network bandwidth usage to the unit CPU usage denoted by $\alpha_C$. Then define $C_h$ as the total capacity of vCPUs for Host $h$. The remaining number of vCPUs, $R_C(h)$, can be calculated by

$$R_C(h) = C_h - \sum_{v_h} C(v_h) - \alpha_C \left[ \sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h) \right]. \tag{12}$$

Let $w_C(h)$ stand for the unit cost of consuming the CPU resource of $h$. We model $w_C(h)$ to be inversely proportional to the remaining vCPUs with constant of proportionality $W_C$. Define the CPU cost of $h$ as $S_C(h)$. We have

$$S_C(h) = \sum_{v_h} C(v_h) w_C(h) = \sum_{v_h} \frac{C(v_h) W_C}{R_C(h) + \delta}, \tag{13}$$

where $\delta$ is a small positive number to avoid dividing by zero.

Similar to the CPU cost, define the amount of memory needed by $v$ as $M(v)$. It can be divided into two parts. One fixed part is to complete its own tasks, denoted by $M_v$. The other part is to coordinate with other VMs. Let the amount of memory required to coordinate with other VMs be proportional to the number of VMs to communicate with constant $\gamma_M$. We have $M(v) = M_v + \gamma_M(N_V - 1)$.

Let the conversion ratio from the intra-host unit network bandwidth usage to the unit memory usage denoted by $\alpha_M$. Then define $M_h$ as the total amount of Memory for Host $h$. The remaining memory, $R_M(h)$, can be calculated by

$$R_M(h) = M_h - \sum_{v_h} M(v_h) - \alpha_M \left[ \sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h) \right]. \tag{14}$$

Let $w_M(h)$ stand for the unit cost of consuming the memory resource of $h$. We model $w_M(h)$ to be inversely proportional to the remaining memory with constant of proportionality $W_M$. Define the memory cost of $h$ as $S_M(h)$. We have

$$S_M(h) = \sum_{v_h} M(v) w_M(h) = \sum_{v_h} \frac{M(v_h) W_M}{R_M(h) + \delta}, \tag{15}$$

where $\delta$ is a small positive number to avoid dividing by zero.

## 2.5 | Cloud Costs

For a mobile edge application with SFC where no host and link combination would be able to meet the requirements, the remote cloud can become an option. For the simplicity of our discussion, we give constant unit costs of CPU, memory, and bandwidth as $w_C(c)$, $w_M(c)$ and $w_B(c)$.

Define the CPU, memory and bandwidth costs as $S_C(c)$, $S_M(c)$, and $S_B(c)$. The total cost for deploying VMs on the cloud is denoted by $S(c)$. Compared to VM deployments on hosts on the mobile edge, apparently, deployment VMs on the cloud has significantly higher bandwidth cost. The total bandwidth from the mobile edge to the cloud is defined by $B_c$, which will be the bottleneck of cloud-based VM deployments if more VMs are placed on the cloud. The total cloud cost, denoted by $S(c)$, is then

$$\begin{aligned} S(c) &= S_C(c) + S_M(c) + S_B(c) \\ &= \sum_{v_c} C(v) w_C(c) + M(v) w_M(c) + B(v) w_B(c). \end{aligned} \tag{16}$$

## 2.6 | Stochastic Programming Formulation

Stochastic programming is an approach for modeling optimization problems that involve uncertainty. It provides a solution by eliminating uncertainty and characterizing it using probability distributions. There are in general two types of stochastic programming: probabilistic constraints and recourse problems. In probabilistic constraints, the optimization problems are deterministic with some constraints containing probability distribution functions. In recourse problems, the objectives are to optimize the results in average, which typically has two stages, where we make some decisions in the first stage and we make further decisions in the second stage to avoid the constraints of the problem becoming infeasible after seeing a realization of the stochastic elements. In this dissertation, our stochastic programming problems fall in the first type in the sense that some of constraints contain probability functions.

The problem is formulated as a stochastic programming optimization as $P_V$ and $P_H$ used for minimum availability constraints are probability density functions of the random variables. Therefore, the stochastic programming we present is of type probabilistic constraints[19,20].

Regarding its objective, the owner of the mobile edge application aims to minimize the cost when operating MEC services. As discussed in Sections 2.3 and 2.4, we provide three sets of costs, which are $S_B(e_{ij})$, $S_C(h)$ and $S_M(h)$. The optimization is to minimize three types of costs over all hosts and their links.

$$\begin{aligned}
Minimize \quad & S(c) + \sum_h S_C(h) + \sum_h S_M(h) + \sum_{e_{ij}, i\neq j} S_B(e_{ij}) \\
&= \sum_{v_c} C(v)w_C(c) + M(v)w_M(c) + B(v)w_B(c) \\
&+ \sum_{v_h} \frac{\left[C_{v_h} + \gamma_C(N_V - 1)\right] W_C}{R_C(h) + \delta} \\
&+ \sum_{v_h} \frac{\left[M_{v_h} + \gamma_M(N_V - 1)\right] W_M}{R_M(h) + \delta} \\
&+ \sum_{e_{ij}, i\neq j} \frac{B_V^2(e_{ij}) W_B \left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} L(v_{h_i}, v_{h_j})\right]}{B(e_{ij})\left[B(e_{ij}) - B_V(e_{ij})\right] a_{ij} + \delta}
\end{aligned} \tag{17}$$

$$w.r.t. \quad x_{vh}$$

$$s.t. \quad B(e_{ij}) \geq \sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} B(v_{h_i}, v_{h_j}) \tag{18}$$

$$\sum_{v_c, v_h} B(v_c, v_h) \leq B_c \tag{19}$$

$$\begin{aligned} C_h \geq & \sum_{v_h} \left[C_{v_h} + \gamma_C(N_V - 1)\right] \\ & + \alpha_C \sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h) \end{aligned} \tag{20}$$

$$\begin{aligned} M_h \geq & \sum_{v_h} \left[M_{v_h} + \gamma_M(N_V - 1)\right] \\ & + \alpha_M \sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h) \end{aligned} \tag{21}$$

$$\sum_{\bar{a}, \hat{a}=y}^{N_V} p_{\bar{a}} \geq 1 - \eta \tag{22}$$

## Remarks

- ⋆ Function (17) is the objective function. It targets to minimize the total cost. The host placement policy tends to find a sweet point minimizing the cost by using less hosts, while not exhausting them.
- ⋆ Constraint (18) is the link bandwidth capacity bounds between each two hosts. Traffic transmitted between any two hosts $h_i$ and $h_j$ must not exceed the corresponding bandwidth capacity $B(e_{ij})$.

- ⋆ Constraint (19) is the link bandwidth capacity bound from the mobile edge to the remote cloud. The total bandwidth consumption between the VMs at the edge and those on the cloud cannot exceed $B_c$, which is the bandwidth between the edge and the cloud.
- ⋆ Constraints (20) and (21) are the CPU and memory capacity bounds for each host. The CPU and memory used by VMs coordinating with each other and by intra-host communications must not exceed $C_h$ and $M_h$.
- ⋆ Constraint (22) sets the bottom line of the number of host available for keeping the mobile edge application available, i.e., the probability of at least $N_m$ VMs in service must be greater than or equal to $1 - \eta$. This constraint would lead to anti-affinity rules enforced among at least part of the VMs to ensure at least the required number of hosts are used.

**Algorithm 1** EdgePlace VM Sorting Algorithm to Determine the Ordering for Future Placement

```
1:  function SortVmList(vm_list)
2:      sort vm_list by their sequence in the service function chain ascending
3:      for all VMs with the same sequence number do
4:          sort VMs by bandwidth requirements descending
5:      end for
6:      for all VM in vm_list do
7:          if there are multiple VMs with the same number of connected VMs then
8:              sort these VMs by bandwidth requirements descending
9:              if there are multiple VMs with the same bandwidth requirements then
10:                 sort these VMs by their neighbors' total number of connected VMs
11:             end if
12:         end if
13:     end for
14:     return sorted vm_list
15: end function
```

## 2.7 | Scalability and the EdgePlace Algorithm

The formulation presented is one of the stochastic programming problems, where $P_H$ and $P_V$ are used in Constraint (22). Combining Equation (3), we found that Constraint (22) will contain exponential functions depending on the value of $a_h$. Therefore, the stochastic programming presented above is intractable. As the problem scales, it may not be computationally feasible to solve it. To apply our model to real-world scenarios, we develop a heuristic algorithm called EdgePlace to achieve suboptimal results by applying a hybrid strategy of best-fit and first-fit decreasing algorithm.

### 2.7.1 | Processing Order of VMs

The EdgePlace algorithm will first process VMs with the most links to other VMs. These VMs tend to cause more inter-host traffic if migrated. Therefore, the algorithm tries to put VMs linked to each other on the same host. Even if the host is down, these VMs can be migrated together to another host without generating extra inter-host traffic. EdgePlace follows the ordering to sort VM as below, also shown in Algorithm 1.

- ⋆ Sort all VMs by the number of connected VMs (neighbors) descending.
- ⋆ Sort VMs with the same number of connected VMs by their bandwidth requirement descending.
- ⋆ Sort VMs with the same bandwidth requirement by the sum of connected VMs of their neighbors.

### 2.7.2 | Affinity first with service chain consideration

As Algorithm 2 shows, when deploying a mobile edge application VM, all hosts are sorted based on their remaining resources as the first step. When the availability requirement is met, EdgePlace tries affinity first: the host with the most existing VMs deployed is attempted first to minimize inter-host traffic of the mobile edge application. If all constraints are satisfied, the host will be chosen to deploy the VM.

When a mobile edge application service is chained, traffic streams flowing through VMs will be directional. Unlike clustered VMs that can exchange data with each other, it doesn't save inter-host bandwidth if we put together two non-adjacent VMs in a service chain. In fact, it may cause traffic to go back and forth and worsen the situation. To reduce inter-host traffic by SFC, EdgePlace takes into consideration the sequence of VMs, and try to put adjacent VMs on the same host when the resource level permits. When moving placement decisions to another host, the algorithm will not consider previously-chosen hosts anymore, to avoid repeating traffic between hosts.

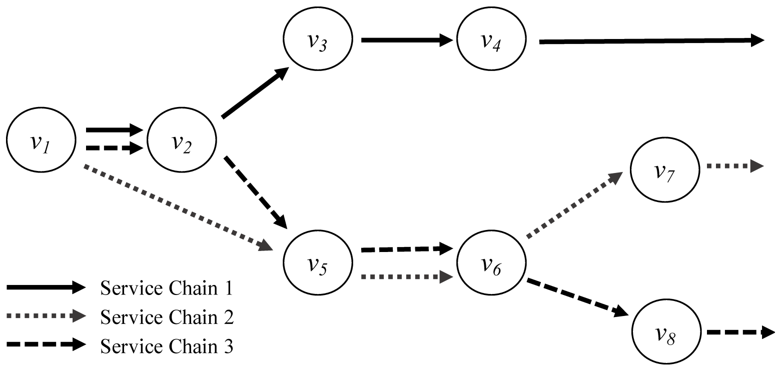

**FIGURE 5 A service graph consisting of 3 service chains.**

Fig. 5 is an example of 3 service chains established on 8 VMs. Based on our algorithm, we fist determine the sequence of the VMs in the service chains. Note that if a VM is shared by more than one service chain, we take the smallest sequence. For instance, while $v_5$ is the third VM for Service Chain 3, it is also the second VM for Service Chain 2. Therefore, its sequence is 2. Based on the sequence of each VM, their placement will be determined by the algorithm accordingly. When the sequence of multiple VMs are the same, the priority of placement decision for those VMs will then be sorted by the bandwidth consumption.

### 2.7.3 | Anti-affinity by bandwidth cost

Otherwise, EdgePlace will try the host with the lowest link bandwidth costs given the VM to be deployed would cause inter-host traffic. At each step to calculate the potential inter-host bandwidth cost, the algorithm will only calculate the added bandwidth cost to VMs already deployed. The VMs yet to be deployed will not be considered until the algorithm reaches iterations to process them. The step will be repeated until it has tried all hosts or it has found the first valid host to deploy the VM. As one of the constraints, the availability change of the mobile edge application for the placement selection is tracked. To deploy all VMs, the worst time complexity of the algorithm is $O(N_V^2 \log N_H)$, where sorting the hosts takes $O(\log N_H)$ and iterating all VMs on all hosts takes $O(N_V^2)$.

### 2.7.4 | Practical use of EdgePlace Algorithm

As we have clarified the processing order of the VM, it will be quite feasible to run EdgePlace Algorithm on a MEC base station for host selection when the availability of the mobile edge applications needs to be guaranteed. A practical example would be an OpenStack controller located on a MEC base station with a cluster of computing hosts. The EdgePlace Algorithm can be implemented as a filter to be called by Nova scheduler. Whenever a request arrives to deploy a mobile edge application, our algorithm can be executed to find the first host that meets both the resource requirements and the availability constraint.

**Algorithm 2** EdgePlace Host Selection Algorithm

```
1: function EdgePlace($v, H$)
2:     host_list ← $empty$
3:     Add $h_1$ with most deployed VMs to host_list
4:     Sort other host by remaining bandwidth to all deployed VMs descending and add to host_list
5:     for all $h \in$ host_list do
6:         $R_C(h) \leftarrow C_h$
7:         $R_M(h) \leftarrow M_h$
8:         for all $v_h$ do
9:             $R_C(h) \leftarrow R_C(h) - C_{v_h} - \gamma_C(N_V - 1)$
10:            $R_M(h) \leftarrow R_M(h) - M_{v_h} - \gamma_M(N_V - 1)$
11:            if $v_h \neq v$ then
12:                $R_C(h) \leftarrow R_C(h) - \alpha_C B(v, v_h)$
13:                $R_M(h) \leftarrow R_M(h) - \alpha_M B(v, v_h)$
14:            end if
15:        end for
16:        if $R_C(h) < 0$ or $R_M(h) < 0$ then
17:            Skip $h$ and check the next host
18:        end if
19:        for all $v_j$ and $h \neq j$ do
20:            $R_B(e_{hj}) \leftarrow R_B(e_{hj}) - B(v, v_j)$
21:            if $R_B(e_{hj}) < 0$ then
22:                Skip $h$ and check the next host
23:            end if
24:        end for
25:        Calculate $p_y$ assuming $v$ on $h$
26:        if $p_y < 1 - \eta$ then
27:            Skip $h$ and check the next host
28:        end if
29:        cost ← $S_C(h) + S_M(h) + S_B(e_{ij})$
30:        return $h$
31:    end for
32: end function
```

## 3 | NUMERICAL RESULTS

In this section, we illustrate the numerical results of availability changes with different scenarios of mobile edge application deployments.

### 3.1 | Assumptions

To clearly demonstrate the focused trends, the following assumptions are made to simplify the modeling of the problem without losing generality. We first discuss the placement selection of the same mobile edge application with elasticity. This means variable numbers of VMs can be deployed for the application, but all VMs are of the same type and require the same amount of resources.

1. The unit costs of the CPU and memory across all hosts are the same. So are costs of network bandwidth across all links among hosts.
2. One mobile edge application includes the same type of VMs with the same CPU, memory and network bandwidth requirements.

3. A request from the user will be processed by one VM, while the VM may communicate with other VMs to exchange information.

## 3.2 | Parameters

With the assumptions above, we choose parameters for our placement model to evaluate the performance and the facts under different circumstances.

### 3.2.1 | Hardware Requirement Profile

The hardware requirement profile of a mobile edge application describe its resource needs. In this paper, it refers to the number of vCPUs, amount of memory and bandwidth. We define 10 types of hardware requirement profiles from F1 to F10, each with its unique requirements for the three resources. Based on typical VNF hardware requirements[21,22,23,24], we believe these 10 flavors are valid and can cover the cases for general VNFs. In order to further justify that these flavors are representative, we pick 3 of them for their potential actual scenarios. The first example is for F2, which has 2 vCPUs, 2048 MB of memory, 2 Gbps bandwidth and 100 ms maximum latency configured. This configuration is good for a financial mobile edge application that is not too CPU- or memory-intensive but has some network transmission requirements and is strict on networking latency. Another example can be F6 with 4 vCPUs, 4096 MB of memory, 4 Gbps bandwidth and 1000 ms maximum latency. A typical example of mobile edge application that fits F6 can be a virtual router like Cisco CSR1000V[21] as it requires moderate amount of resources and can tolerate a certain level of networking latency. The last example is F10 with 8 vCPUs, 8192 MB of memory, 8 Gbps bandwidth and 10000 ms maximum latency. This flavor has the most resource requirements but the lowest networking latency requirement. A high-throughput data storage application will be a good match for F10 as it needs to write large amount of data constantly but does not require the storage process to complete immediately. Above all, the 10 flavors we choose below are believed to cover a wide range of real-world use cases.

**TABLE 2** Pre-defined hardware requirement profiles for simulation.

| Name | $C_v$ | $M_v$ | $B_v$ | Max Latency(ms) |
|---|---|---|---|---|
| F1 | 1 vCPU | 1024 MB | 1 Gbps | 1000 ms |
| F2 | 2 vCPUs | 2048 MB | 2 Gbps | 100 ms |
| F3 | 2 vCPUs | 2048 MB | 2 Gbps | 1000 ms |
| F4 | 2 vCPUs | 2048 MB | 2 Gbps | 10000 ms |
| F5 | 4 vCPUs | 4096 MB | 4 Gbps | 100 ms |
| F6 | 4 vCPUs | 4096 MB | 4 Gbps | 1000 ms |
| F7 | 4 vCPUs | 4096 MB | 4 Gbps | 10000 ms |
| F8 | 8 vCPUs | 8192 MB | 8 Gbps | 100 ms |
| F9 | 8 vCPUs | 8192 MB | 8 Gbps | 1000 ms |
| F10 | 8 vCPUs | 8192 MB | 8 Gbps | 10000 ms |

We choose certain parameters in our model as constants, while others as variables, shown as Table 3 , where values are specified for constants, and variables are marked as *var*. For the constants, it is reasonable to assume that they are fixed for a specific mobile edge application deployment scenario. For the variables, they can change because scaling and SLA may change due to the nature of MEC. We will further discuss the variables in different sets of experiments.

Note that for the value of $N_V$, in this paper, we choose values varying from 6 to 20. These values chosen are valid cases enabled by VNF scaling[25].

**TABLE 3** Parameters for Availability-aware Host Selection

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $C_v$ | 2 vCPUs | $M_v$ | 2048 MB |
| $\alpha_C$ | 10% | $\alpha_M$ | 10% |
| $M_h$ | 102400 MB | $C_h$ | 100 vCPUs |
| $B(v, v')$ | 20 Mb | $\gamma_C$ | 10% |
| $\gamma_M$ | 0.1% | $B(e_{ij})$ | 10000 Mb |
| $P_V$ | 90% | $P_H$ | 90% |
| $T_h$ | 5 ms | $T_c$ | 100 ms |
| $k_{th}$ | 1.1 | $k_{tc}$ | 1.5 |
| $\delta$ | 0.1 | $y$ | *var* |
| $W_C$ | *var* | $W_M$ | *var* |
| $W_B$ | *var* | $\eta$ | *var* |
| $N_V$ | *var* | $N_H$ | *var* |

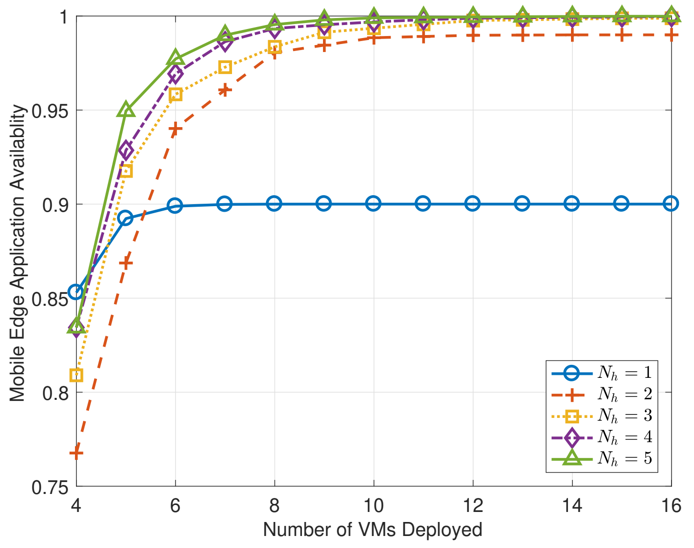

**FIGURE 6 Availability of the mobile edge application with different numbers of VMs deployed on various numbers of mobile edge hosts.**

## 3.3 | Availability Impact from Number of Instances and Hosts

We first discuss the mobile application availability trends with various numbers of instances deployed on certain numbers of hosts. The discussion assumes constants $W_C = W_M = W_B = 5$, and $y = 3$. Also, $\eta$ is set to be 1 such that no minimum availability is required. This will help observe the trends of availability changes.

Fig. 6 demonstrates availability changes with the number of instances deployed ranging from 4 to 16, and the number of hosts from 1 to 5. From the results, we can clearly see that increasing the number of backup VMs can improve the availability of the mobile edge application. However, there is a ceiling of the availability by only increasing the number of VMs due to host availability. When $N_H = 1$, that maximum availability cannot exceed 0.9. To meet the minimum availability, there must be enough hosts to keep the theoretical availability above $1 - \eta$.

Furthermore, Fig. 7 shows the negative impact from $y$, the minimum number of VMs required by the mobile edge application. With the $N_V = 10$ as a constant, having a larger $y$ would lower the availability of the application. Considering to increase the number of VMs as well as hosts may be necessary when demand is high.

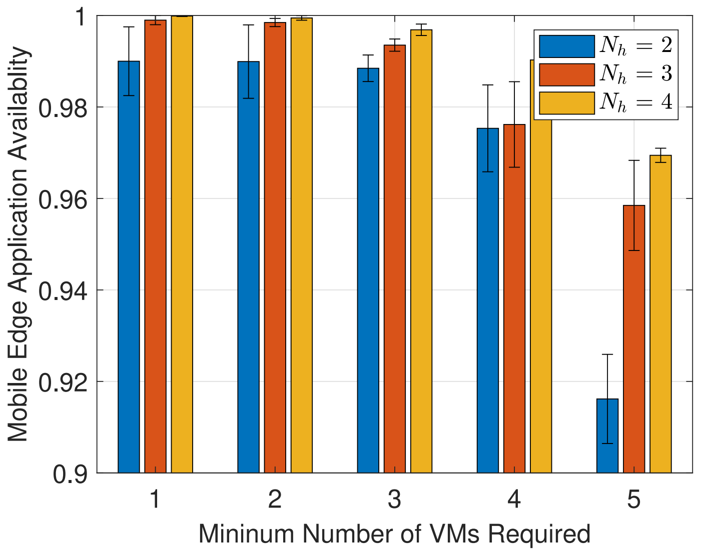

**FIGURE 7 Availability of the mobile edge application with varying minimum number of VMs required and different numbers of mobile edge hosts. Confidence level is 95%.**

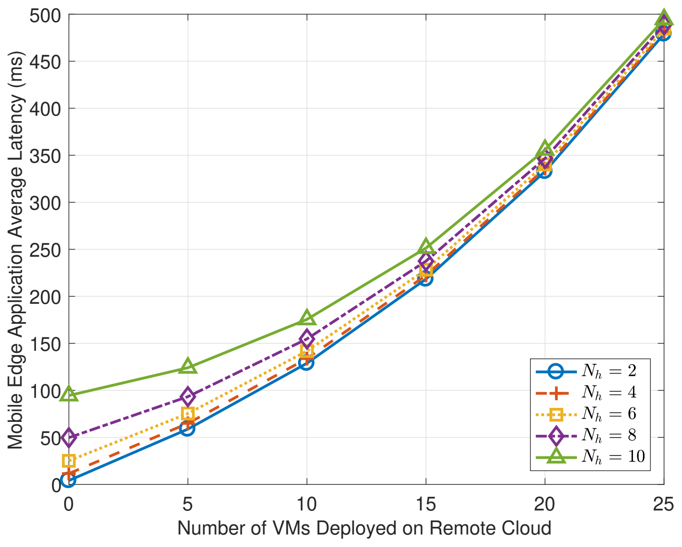

**FIGURE 8 Average latency of the mobile edge application with different numbers of VMs deployed on remote cloud and various numbers of hosts.**

## 3.4 | Latency

Assume the latency of a link between each two MEC hosts to be $T_h$, while the link between a MEC host and the remote cloud to be $T_c$. As more VMs deployed for one mobile edge application, traffic increases among them to keep them in sync and to coordinate with each other. We set $T_h = 6ms$, and $T_c = 200ms$. The latencies caused by inter-host traffic and host-cloud traffic are shown in Fig. 8 . The results have demonstrated an increase of latency as the number of hosts grows for deploying the mobile edge application. Depending on the definition of the application and its maximum latency allowed, we can determine the maximum number of VMs offloaded to the cloud under different numbers of mobile edge hosts.

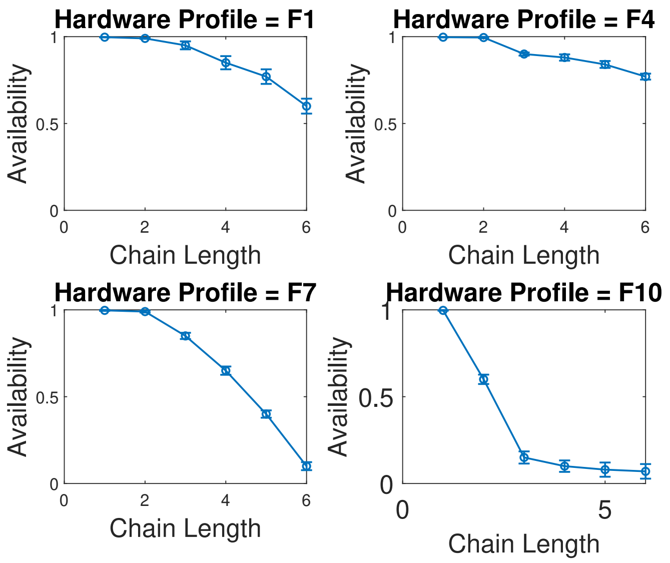

**FIGURE 9 Availability of a mobile edge application consisting of 6 VMs, each with different hardware requirement profiles (F1, F4, F7, and F10). Each sub figure shows the availability dropping as the length of the service chain increases. Confidence level is 95%.**

## 3.5 | Availability Impact from SFC

The portability of a mobile edge application due to SFC can affect the application availability. We compare the availability among mobile edge applications with different levels of SFC. We first define a mobile edge application consisting of 6 VMs with different level of chaining, starting from no chaining, to the first certain number of VMs chained, and to all 6 VMs chained. In general, SFC will bring negative impact to the availability, because it sets more constraints when placing VMs, making it more difficult to find valid place for hosting the VMs. Four pre-defined hardware requirement profiles from Table 2 , namely F1, F4, F7, and F10, are chosen for creating VMs for the target mobile edge application. Fig. 9 shows availability changing trends under 4 hardware requirement profiles. From the results, we find the availability plunges faster when VMs have higher hardware requirements. This is due to the increasing challenge to find proper host combinations to deploy the application with SFC.

## 3.6 | Distribution of VMs with Different Bandwidth Costs

The impact of $W_B$, which can be considered as the network bandwidth price index, is evaluated in this section. We set up $N_H = 10, N_V = 20, y = 2, W_C = W_M = 1,$ and $\eta = 0.3\%$. We choose various values of $W_B$ and the results of VM placements under each value of $W_B$ on 10 hosts from $H01$ to $H10$, as shown in Fig. 10 . From the results, it can be learned that higher price of the network bandwidth will cause more concentrated placement of the VMs. When the host networks become congested, EdgePlace would put VMs on less hosts to limit inter-host traffic, with the trade-off lowering the availability of the mobile edge application. When the hosts used are reduced to 3 with a 9-9-2 VM distribution, the availability is close to the bottom line of 99.73%. If $W_B$ continues to hike, choosing a 10-9-1 VM distribution would achieve the lowest cost, but would violate the minimum availability requirement. Therefore, the 9-9-2 VM distribution is the best placement decision when $W_B \geq 45$.

## 4 | RELATED WORK

The telecommunication industry has been shifting focus for years to providing even higher speed and lower latency connections to UE, with technologies like 5G and MEC. There has been much work done in this area from different perspectives.

Farris et al.[26] presented a solution which leveraged proactive service replication for stateless applications in MEC environment. Compared to classic reactive service migration, their solution significantly reduced the time of service migration between different cloudlets and to meet the latency requirements.

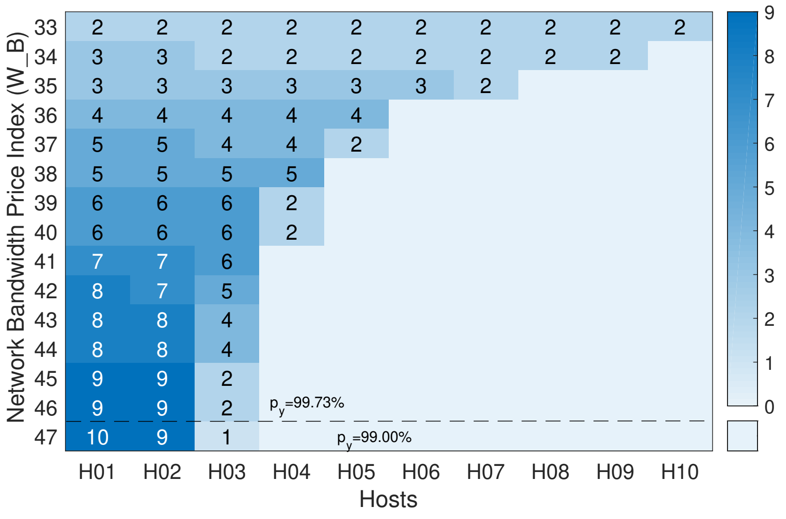

**FIGURE 10 Placement distribution of 20 VMs across 10 hosts with $N_H = 10, N_V = 20, y = 2, W_C = W_M = 1,$ and $\eta =$ 0.3%. Empty cell means no VM is deployed on the host with a certain $W_B$.**

Yaseen et al.[27] proposed a model providing a global monitoring capability for tracing moving sensors and detecting selective forwarding attacks. The model leveraged fog computing infrastructure and software defined systems to secure MWSNs against multiple types of attacks.

Regarding IoT device security in the MEC context, a methodology was developed by Pacheco et al.[28] with anomaly behavior analysis to detect when a sensor has been compromised and used to provide misinformation.

Fog nodes for smart factories were discussed by de Brito et al.[29] to show the advantages of having a programmable Fog Node supported by an orchestration system. Their research extended a standard-compliant machine-to-machine communication architecture to support container-based orchestration mechanisms to enable cyber-physical systems to be programmable, autonomous, and to communicate peer-to-peer.

ReCAP[30] was proposed as a distributed CAPTCHA service at the edge of the network to handle server overload caused by application layer-based DDoS attacks or flash crowd events. The main goal of ReCAP is to filter attack traffic in case of a DDoS attack event and to provide users with basic information during a flash crowd event.

Another framework called SIMDOM[31] was presented for for single instruction multiple data (SIMD) instruction translation and offloading for mobile devices in cloud and edge environments. The SIMDOM framework reduced the execution overhead of migrated vectorized multimedia application by using vector-to-vector instruction mappings. The framework mapped and translated ARM SIMD intrinsic instructions to x86 SIMD intrinsic instructions such that an application programmed for the mobile platform can be executed on the cloud server without any modification. The offload decision was based on inputs from the device energy, network, and application profilers.

Orsini et al. developed CloudAware[32] as a mobile cloud computing/mobile edge computing middleware that supported automated context-aware self-adaptation techniques for offloading. The system utilized compositional adaptation and sensor-based reasoning to allow a flexible adaptation to current as well as future context states that can hardly be foreseen by developers. In particular, connectivity and execution predictions were used that support the efficiency of offloading decision in MCC as well as MEC scenarios.

Garcia-Perez et al.[33] explored 2 ways of using these new technologies to reduce the latency in Long-Term Evolution (LTE) networks. The first solution, called Fog Gateway, was based on the interception of the packets in the tunnel at the eNB and their redirection to local servers running the fog services. This solution is fully compliant with the current LTE architecture and only requires new components. The second solution, called General Packet Radio Service Tunneling Protocol Gateway (GTP), was based on splitting the eNB's functionality to avoid unnecessary GTP encapsulation of the packets geared toward the fog services.

While the models and frameworks presented above focused on MEC and IoT and some discussed offloading and latency, none of them modeled the availability of mobile edge applications and attempted to use placement policies to maintain the application availability. Our research has filled the gap of availability consideration and can be combined with offloading frameworks such as proactive service replication[26] to complement the existing proposed systems.

The ETSI architecture of MEC[34] leverages the existing NFV framework[35] to achieve dynamic and fast MEC service provisioning. To our best knowledge, there has not been much research on mobile edge application placement. Main references of related work are still from the NFV world. OpenFog[9] is another MEC architecture particularly suited to IoT systems. The OpenFog architecture serves a specific subset of business problems that cannot be successfully implemented using "cloud only" based architectures or relying solely on intelligent edge devices. The goal of the OpenFog architecture is to facilitate deployments which highlight interoperability, performance, security, scalability, programmability, reliability, availability, serviceability, and agility. Considering integration to the two standards above, or other architectures for MEC, EdgePlace serves as a generic model to deploy application on same or different hosts with availability awareness. Therefore, it is not tightly coupled to a single architecture.

NFV faces new research challenges due to the new features it introduces, such as flexibility of deploying network functions and similar pricing model to cloud computing[36]. Among many other research challenges, mobile edge application placement policies have been studied with various focuses and is typically modeled as a resource allocation problem. Deployed without consideration of optimal resource allocation, real-world mobile edge application deployments are found inefficient to utilize resources through instrumentation effort[37]. The goal of the resource optimization is to find on the best physical resources (servers) to place network functions. Such problems have been formulated and studied by Basta et al.[38].

For the VM placement issue in a cloud data center, a comprehensive study of the VM placement and consolidation techniques used in cloud was presented by Usmani et al.[39]. The VM placement problem and various approaches were reviewed. The placement techniques were classified as constraint programming, bin packing, stochastic integer programming and genetic algorithm. Bellur et al.[40] and Khosravi et al.[41] focused on VM allocation in data centers as well. With an optimal technique, Bellur et al.[40] aimed to minimize the number of required VMs, with considerations of each VM's resource limitation. A VM placement algorithm was proposed[41] with the objective to minimize carbon footprint. An NFV traffic steering problem was discussed by Zhu et al.[42], where the limited resources were taken as constraints to determine the lowest cost and to steer traffic accordingly.

In mobile edge application networks, the virtual resources include virtual machines, which execute either virtual routers or virtual service elements. To address the challenges related to the management and orchestration of virtual resources, Clayman et al.[43] described an architecture based on an orchestrator that enabled automated placement of virtual network and processing resources across the physical resources of the network. Least used placement, N-at-a-time placement and Least Busy placement algorithms were presented.

Similarly, VNF-P[44] focused on the mobile edge application deployments by presenting a model for resource allocation in NFV networks, while also considering the difference between service requests and VM requests. It addressed the scenario where part of the services may be provided by dedicated physical hardware, with the other part using virtualized service instances. A basic model using linear programing was defined for NFV resource allocation, with the constraints of server capacity and the number of instances on a node. The objective of the model is to minimize the number of user servers. To add network-awareness to the model, constraints related to the request flow were added.

Service function chaining (SFC) has been used widely in carrier networks[45], in order to allow configuring network services dynamically without having to change networks at the hardware level. A service graph is used to describe service chains and traffic is routed according to the graph. Use cases leveraging SFC include network functions for packet inspection, traffic optimization, protocol proxies, and value-added services. Network service headers were used[46] to provide data-plane information for constructing topological-independent services. A software-defined architecture to enable SFC was proposed by Blendin et al.[47] leveraging OpenFlow with discussion of functionalities, challenges, testbeds, and other aspects implementation-wise. A heuristic resource allocation algorithm was proposed for VNF chains called CoordVNF[48]. Substrate nodes with enough resource to deploy VNF were explored and the subsequent nodes were checked iteratively. If any of the substrate nodes cannot embed a VNF in the chain, backtracking was performed.

When attempting to achieve various goals, existing work does not appear to consider the combination of low cost and high availability together with practical placement policies, which are affinity and anti-affinity, In this paper, these factors are weighed in, and the strategy is ready for use by real-world mobile edge application scenarios.

## 5 | CONCLUSIONS AND FUTURE WORK

In this paper, we have formulated a mobile edge application placement problem and presented a heuristic algorithm called EdgePlace. Our work promotes reducing the cost by introducing extra mobile edge hosts and balancing the workload of the hosts. Rather than overloaded, expensive hosts, VMs are deployed on less busy hosts to achieve lower cost. Meanwhile, the availability of the mobile edge application has also been profoundly improved as a result of leveraging multiple hosts.

Our future work includes the following aspects:

1. Design an architecture supporting running EdgePlace Algorithm in the real MEC environment. This will enable the access to practical experimental data for evaluation. We plan to implement EdgePlace Algorithm as a filter in OpenStack which share the similar architecture in previous research done by us[49]. With a custom filter implementing EdgePlace, we will be able to use the algorithm for making placement decisions when OpenStack is used at MEC base stations.

2. Consider mobile edge applications with combinations of different NFs and with service chaining. Also, the mobility of the cloud needs to be considered as at the mobile edge, base stations along with the compute hosts can become mobile and loaded onto vehicles.

## Financial disclosure

None reported.

## Conflict of interest

The authors declare no potential conflict of interests.

## References

1. Satyanarayanan Mahadev. The Emergence of Edge Computing. *Computer.* 2017;50(1):30-39.

2. Garcia Lopez Pedro, Montresor Alberto, Epema Dick, et al. Edge-centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.*. 2015;45(5):37–42.

3. Hu Yun Chao, Patel Milan, Sabella Dario, Sprecher Nurit, Young Valerie. Mobile Edge Computing - A key Technology Towards 5G. *ETSI White Paper.* 2015;11.

4. Jaquith Waldo. The New Trend of Regional Data Centers https://usopendata.org/2015/03/25/rdc/[Online; accessed Mar-04-2017]; .

5. Satyanarayanan Mahadev, Bahl Paramvir, Caceres Ramón, Davies Nigel. The Case for VM-based Cloudlets in Mobile Computing. *IEEE pervasive Computing.* 2009;8(4).

6. Bonomi Flavio, Milito Rodolfo, Zhu Jiang, Addepalli Sateesh. Fog Computing and Its Role in the Internet of Things. In: :13–16ACM; 2012.

7. Martins Joao, Ahmed Mohamed, Raiciu Costin, et al. ClickOS and the art of network function virtualization. In: :459–473USENIX Association; 2014.

8. Han Bo, Gopalakrishnan Vijay, Ji Lusheng, Lee Seungjoon. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine.* 2015;53(2):90–97.

9. OpenFog Consortium . OpenFog Architecture Overview - White Paper. 2016;OPFWP001.0216.

10. Liu H., Eldarrat F., Alqahtani H., Reznik A., Foy X., Zhang Y.. Mobile Edge Cloud System: Architectures, Challenges, and Approaches. *IEEE Systems Journal.* 2017;PP(99):1-14.

11. Kevin Brown Schneider Electric. Edge computing needs reliability http://www.datacenterdynamics.com/content-tracks/power-cooling/edge-computing-needs-reliability/97587.fullarticle[Online; accessed Mar-24-2017]; .

12. Beck Michael Till, Werner Martin, Feld Sebastian, Schimper S. Mobile edge computing: A taxonomy. In: Citeseer; 2014.

13. Jammal M., Kanso A., Shami A.. High availability-aware optimization digest for applications deployment in cloud. In: :6822-6828; 2015.

14. Oechsner S., Ripke A.. Flexible support of VNF placement functions in OpenStack. In: :1-6; 2015.

15. Vyas Uchit. Designing Your First Cloud with OpenStack. In: Springer 2016 (pp. 1–17).

16. Halpern Joel M., Pignataro Carlos. Service Function Chaining (SFC) Architecture RFC 76652015.

17. Lee Seungik, Pack Sangheon, Shin Myung-Ki, Browne Rory. Resource management in service chaining. ;.

18. Cassen Alexandre. Keepalived: Health checking for LVS & high availability. *URL http://www.linuxvirtualserver.org*. 2002;.

19. J. E. Beasley . Stochastic programming http://people.brunel.ac.uk/ mastjjb/jeb/or/sp.html[Online; accessed May-8-2018]; 2018.

20. Gaivoronski Alexei A, Lisser Abdel, Lopez Rafael, Xu Hu. Knapsack problem with probability constraints. *Journal of Global Optimization.* 2011;49(3):397–413.

21. Cisco . CSR 1000v requirements https://cisco.com/c/en/us/td/docs/routers/csr1000/release/notes/csr1000v_3Srn.html[Online; accessed May-8-2018]; 2018.

22. Cisco . CSR ASA memory requirements https://cisco.com/c/en/us/products/collateral/security/asa-5500-series-next-generation-firewalls/product_bulletin_c25-586414.html[Online; accessed May-8-2018]; 2018.

23. F5 Networks . Prerequisites for BIG-IP Virtual Edition on ESXi https://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/bigip-ve-setup-vmware-esxi-12-1-0/2.html[Online; accessed May-8-2018]; 2018.

24. AVI Networks . Avi Vantage hardware requirements https://avinetworks.com/docs/17.2/system-requirements-hardware/[Online; accessed May-8-2018]; 2018.

25. OpenStack . VNF scaling https://www.cisco.com/c/en/us/td/docs/routers/csr1000/release/notes/csr1000v_3Srn.html[Online; accessed May-8-2018]; 2018.

26. I. Farris, T. Taleb, H. Flinck, A. Iera. Providing ultra-short latency to user-centric 5G applications at the mobile network edge. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3169. e3169 ett.3169.

27. Qussai Yaseen, Firas Albalas, Yaser Jararwah, Mahmoud Al-Ayyoub. Leveraging fog computing and software defined systems for selective forwarding attacks detection in mobile wireless sensor networks. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3183. e3183 ETT-16-0367.R1.

28. Jesus Pacheco, Salim Hariri. Anomaly behavior analysis for IoT sensors. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3188. e3188 ETT-16-0338.R1.

29. Brito M.S. , S. Hoque, R. Steinke, A. Willner, T. Magedanz. Application of the Fog computing paradigm to Smart Factories and cyber-physical systems. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3184. e3184 ett.3184.

30. T. Al-Hammouri Ahmad, Zaid Al-Ali, Basheer Al-Duwairi. ReCAP: A distributed CAPTCHA service at the edge of the network to handle server overload. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3187. e3187 ett.3187.

31. Junaid Shuja, Abdullah Gani, Kwangman Ko, et al. SIMDOM: A framework for SIMD instruction translation and offloading in heterogeneous mobile architectures. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3174. e3174 ett.3174.

32. Gabriel Orsini, Dirk Bade, Winfried Lamersdorf. CloudAware: Empowering context-aware self-adaptation for mobile applications. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3210. e3210 ett.3210.

33. Augusto Garcia-Perez Cesar, Pedro Merino. Experimental evaluation of fog computing techniques to reduce latency in LTE networks. *Transactions on Emerging Telecommunications Technologies.* ;29(4):e3201. e3201 ett.3201.

34. ETSI Group Specification . Mobile Edge Computing (MEC); Framework and Reference Architecture. *ETSI GS MEC 003, V1.1.1 (2016-03).* ;.

35. ETSI Group Specification . Network Function Virtualisation(NFV); Architectural Framework. *ETSI GS NFV 002, V1.1.1 (2013-10).* ;.

36. Mijumbi Rashid, Serrat Joan, Gorricho Juan-Luis, Bouten Niels, Turck Filip De, Boutaba Raouf. Network Function Virtualization: State-of-the-art and Research Challenges. *CoRR.* 2015;abs/1509.07675.

37. Veitch Paul, McGrath Michael J., Bayon Victor. An instrumentation and analytics framework for optimal and robust NFV deployment. *IEEE Communications Magazine.* 2015;53(2):126–133.

38. Basta Arsany, Kellerer Wolfgang, Hoffmann Marco, Morper Hans Jochen, Hoffmann Klaus. Applying NFV and SDN to LTE mobile core gateways, the functions placement problem. In: :33–38; 2014.

39. Usmani Zoha, Singh Shailendra. A Survey of Virtual Machine Placement Techniques in a Cloud Data Center. *Procedia Computer Science.* 2016;78:491 - 498.

40. Bellur Umesh, Rao Chetan S., Kumar S. D. Madhu. Optimal Placement Algorithms for Virtual Machines. *CoRR.* 2010;abs/1011.5064.

41. Khosravi Atefeh, Garg Saurabh Kumar, Buyya Rajkumar. Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers. In: :317–328; 2013.

42. Zhu Jiafeng, Huang Changcheng. A universal protocol mechanism for network function virtualization and application-centric traffic steering. In: :257–262; 2015.

43. Clayman Stuart, Maini Elisa, Galis Alex, Manzalini Antonio, Mazzocca Nicola. The dynamic placement of virtual network functions. In: :1–9; 2014.

44. Moens Hendrik, Turck Filip De. VNF-P: A model for efficient placement of virtualized network functions. In: :418–423; 2014.

45. Brown Gabriel. Service Chaining in Carrier Networks. 2015;.

46. Quinn P., Guichard J.. Service Function Chaining: Creating a Service Plane via Network Service Headers. *Computer.* 2014;47(11):38-44.

47. Blendin J., Ruckert J., Leymann N., Schyguda G., Hausheer D.. Position Paper: Software-Defined Network Service Chaining. In: :109-114; 2014.

48. Beck M. T., Botero J. F.. Coordinated Allocation of Service Function Chains. In: :1-6; 2015.

49. Zhu H., Huang C.. IoT-B&B: Edge-Based NFV for IoT Devices with CPE Crowdsourcing. *Wireless Communications and Mobile Computing.* 2018;(3027269).

50. Bala Anju, Chana Inderveer. Fault tolerance-challenges, techniques and implementation in cloud computing. *IJCSI International Journal of Computer Science Issues.* 2012;9(1):1694–0814.

51. Statista . NFV and SDN market size worldwide by region from 2015 to 2019 https://www.statista.com/statistics/461573/sdn-and-nfv-markets-worldwide-by-region/[Online; accessed Oct-27-2016]; .

52. Amazon . Amazon Web Services (AWS) https://aws.amazon.com/[Online; accessed Mar-4-2017]; .

53. Corradi Antonio, Fanelli Mario, Foschini Luca. VM consolidation: A real case based on OpenStack Cloud. *Future Generation Computer Systems.* 2014;32:118–127.

54. Katyal Mayanka, Mishra Atul. A comparative study of load balancing algorithms in cloud computing environment. *arXiv preprint arXiv:1403.6918.* 2014;.

**How to cite this article:** H. Zhu, and C. Huang (2018), EdgeChain: Trustless Multi-vendor Edge Application Placement, *Transactions on Emerging Telecommunications Technologies*, *2017;00:1–6*.