Availability-aware Mobile Edge Application Placement in 5G Networks

He Zhu[†], Changcheng Huang[†]
[†] Department of Systems and Computer Engineering
Carleton University, Ottawa, ON K1S 5B6, Canada
{hzhu, huang}@carleton.ca

***Abstract*—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 a heuristic algorithm. 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 cloud environment.**

***Keywords*—Mobile Edge Computing, 5G, Placement Policy, Stochastic Optimization, Cloud Computing.**

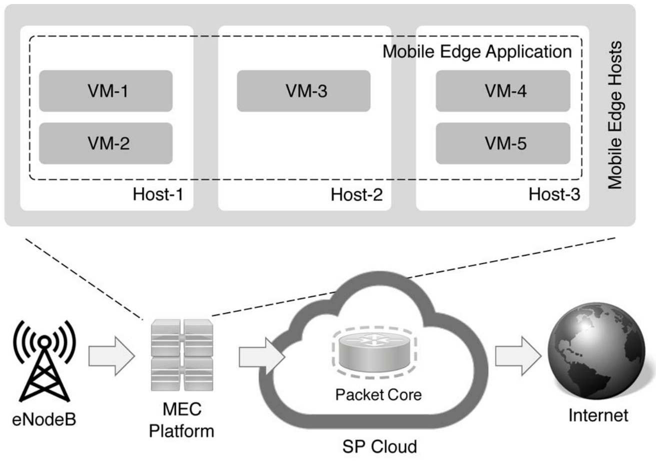


Fig. 1. **A mobile edge application deployment with placement rules. There are five VMs deployed in three groups with each group placed on a separate host. A minimum of three VMs are required for the application. The placement will ensure the application is in service if one host is down.**

I. Introduction

The emerging edge computing is bringing all benefits of cloud computing to mobile devices plus proximity [1], to deliver highly-responsive cloud services at the network edge. As a key technology towards 5G, the mobile edge computing (MEC) architecture proposed by ETSI leverages the widely adopted frameworks of Network Function Virtualization (NFV) [2]. Elastic mobile edge applications are deployed close to the user equipment (UE) with low latency. Both UE application providers and telecommunication service providers (TSPs) can take advantage of MEC to reduce costs and to adjust services with agility based on fast-changing user demands.

A mobile edge application consists of one or more collaborating virtual machines (VMs). It is of paramount importance to maintain the high availability of MEC applications. Compared to centralized datacenters used by public cloud, MEC hosts are heterogeneous with varying computing, storage and networking capabilities [3]. 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 because it serves a smaller group of users. It is unlikely to merit its own security guard or have the same level of redundancy as a larger facility [4]. (ii) The offloading nature of MEC brings higher system complexity which may jeopardize availability [5]. These facts conclude that hosts in MEC are less reliable. When mobile edge applications run in MEC servers, they must be protected from host failure.

To maximize the availability while maintaining costs and latencies at acceptable levels, placement rules, which determine on which computing host each VM is deployed, often come into play to tune the performance and security of a mobile edge application [6]. In practice, placement rules mainly refer to the affinity and anti-affinity rule [7].

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 and require no physical networking infrastructure. Same-host network traffic essentially takes up computing capabilities of the host and has much 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 application would be out of service.

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 of a mobile application out of service. Figure 1 demonstrates an example of a mobile edge application deployment across multiple hosts to increase availability. 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.

With the two placement rules, 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:

- We address the availability concerns of MEC applications even when some hosts are down. We believe at the network edge, hosts are less reliable and availability issues of MEC applications need more attention.
- A cost model is built considering the factors of inter-host traffic and CPU/memory overcommit, to balance the load without causing explosive traffic between hosts.
- We formulate a stochastic programming problem to minimize the cost based on our cost model, while maintaining the availability requirements.
- A heuristic algorithm is developed with suboptimal results as the problem scales. Numerical results show the effectiveness of the algorithm.

We divide the contents in three following sections. Section III formulates the problem. The related work is illustrated in Section II. Then the experimental results are shown in Section IV. Section V concludes the paper.

II. Related Work

The ETSI architecture of MEC [8] leverages the existing NFV framework [9] to achieve dynamic and fast MEC service provisioning. Deployed without consideration of optimal resource allocation, real-world mobile edge application deployments are found inefficient to utilize resources through instrumentation effort [10]. The goal of the resource optimization is to find on the best physical resources (servers) to place network functions. Such problems are formulated and studied in [11].

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 in [12]. The placement techniques were classified as constraint programming, bin packing, stochastic integer programming and genetic algorithm. Research work in [13] and [14] focused on VM allocation in data centers as well. With an optimal technique, [13] aimed to minimize the number of required VMs, with considerations of each VMs resource limitation. A VM placement algorithm was proposed in [14] with the objective to minimize carbon footprint. An NFV traffic steering problem was discussed in [15], 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. Moens et al. [16] 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. 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.

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 NFV scenarios.

III. Problem Formulation

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 $\mathbb{H}$. Assume that there be N_v VMs used by the mobile edge application, denoted by v, and N_h hosts in the MECVI, denoted by h. Define an assigning function x_{vh}, whose value is 1 if VM v is assigned to Host h, 0 otherwise.

$$x_{vh} = \begin{cases} 1, & v \text{ is deployed on } h \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

A. 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, 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 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: even if some VMs are down, there are still more than N_m VMs in service. There must be $N_v \geq N_m \geq 0$.

There are two situations in our discussion that can lead to VM service outage:

- *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.

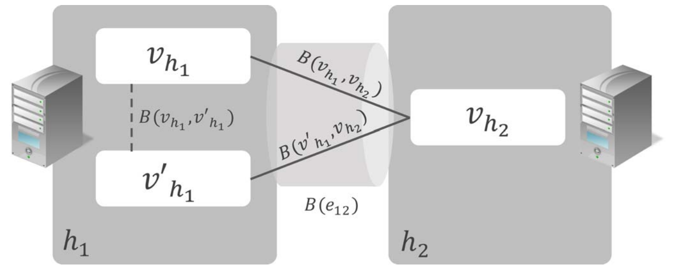


Fig. 2. **Inter-host network bandwidth consumption $B(v_{h_1}, v_{h_2})$ and $B(v'_{h_1}, v_{h_2})$ takes up the bandwidth of e_{12} and incur bandwidth costs. Intra-host network bandwidth consumption $B(v_{h_1}, v'_{h_1})$ consumes extra vCPU and memory of h_1 and incurs vCPU and memory costs.**

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

$$\hat{v}_h = \sum_v x_{vh} \tag{2}$$

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

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

By assumption, different hosts also fail independently, we have

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

Define p_y as the probability that there are at least y VMs available and $\hat{a} = \sum_h a_h$. Then we have

$$p_y \triangleq \Pr\left\{\sum_h A_h \geq y\right\} = \sum_{\bar{a}, \hat{a}=y}^{N_v} p_{\bar{a}} \geq 1 - \eta \tag{5}$$

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

B. Inter-host Network Bandwidth Costs

High availability comes at a cost: extra resources are used for hosting VMs, and extra traffic occurs between VMs working together. 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. Let $B(v_{h_i}, v_{h_j})$ be the total bandwidth (BW) 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 denote the network link between Hosts h_i and h_j as e_{ij} with its total bandwidth capacity defined by $B(e_{ij})$ and residue bandwidth defined by $R_B(e_{ij})$. We calculate the residue bandwidth as following:

$$R_B(e_{ij}) = B(e_{ij}) - \left[\sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} B(v_{h_i}, v_{h_j})\right] \tag{6}$$

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 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 $S_B(e_{ij})$. We have

$$\begin{aligned} S_B(e_{ij}) &= [B(e_{ij}) - R_B(e_{ij})]\, w_B(e_{ij}) \\ &= \frac{W_B\,[B(e_{ij}) - R_B(e_{ij})]}{R_B(e_{ij}) + \delta} \end{aligned} \tag{7}$$

where δ is a small positive number to avoid dividing by zero.

C. 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. Figure 2 shows examples of costs from both inter- and intra-host traffic between two VMs.

Define the number of vCPUs and the amount of memory required by v as $C(v)$ and $M(v)$, each of which can be divided into two parts: one fixed part is to complete v's own tasks, denoted by C_v and M_v. The other part is to coordinate with other VMs. Let the number of vCPUs and the amount of memory required to coordinate with other VMs be proportional to the number of VMs to communicate with constant γ_C and γ_M. We have $C(v) = C_v + \gamma_C(N_v - 1)$ and $M(v) = M_v + \gamma_M(N_v - 1)$.

Let the conversion ratio from the intra-host unit network bandwidth usage to the unit CPU and memory usage be denoted by α_C and α_M, respectively. Then define C_h and M_h as the total capacity of vCPUs and memory for Host h. The remaining number of vCPUs, $R_C(h)$, and the remaining amount of memory, $R_M(h)$, can be calculated by

$$R_C(h) = C_h - \sum_{v_h} C(v_h) - \alpha_C \left[\sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h)\right] \tag{8}$$

$$R_M(h) = M_h - \sum_{v_h} M(v_h) - \alpha_M \left[\sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h)\right] \tag{9}$$

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

$$S_C(h) = \sum_{v_h} C(v_h) w_C(h) = \sum_{v_h} \frac{C(v_h) W_C}{R_C(h) + \delta} \tag{10}$$

$$S_M(h) = \sum_{v_h} M(v) w_M(h) = \sum_{v_h} \frac{M(v_h) W_M}{R_M(h) + \delta} \tag{11}$$

where δ is a small positive number to avoid dividing by zero.

D. Stochastic Programming Formulation

The problem is formulated as a stochastic programming optimization. Regarding its objective, the owner of the mobile edge application aims to minimize the cost when operating MEC services. As discussed in Sections III-B and III-C, 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.

$$\begin{aligned} Minimize \sum_h S_C(h) + \sum_h S_M(h) + \sum_{e_{ij}, i \neq j} S_B(e_{ij}) \\ = \sum_{v_h} \frac{[C_{v_h} + \gamma_C (N_v - 1)] W_C}{R_C(h) + \delta} \\ + \sum_{v_h} \frac{[M_{v_h} + \gamma_M (N_v - 1)] W_M}{R_M(h) + \delta} \\ + \sum_{e_{ij}, i \neq j} \frac{W_B [B(e_{ij}) - R_B(e_{ij})]}{R_B(e_{ij}) + \delta} \end{aligned} \tag{12}$$

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

$$s.t. \quad B(e_{ij}) \geq \sum_{v_{h_i}, v_{h_j}, h_i \neq h_j} B(v_{h_i}, v_{h_j}) \tag{13}$$

$$\begin{aligned} C_h \geq \sum_{v_h} [C_{v_h} + \gamma_C (N_v - 1)] \\ + \alpha_C \sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h) \end{aligned} \tag{14}$$

$$\begin{aligned} M_h \geq \sum_{v_h} [M_{v_h} + \gamma_M (N_v - 1)] \\ + \alpha_M \sum_{v_h, v'_h, v_h \neq v'_h} B(v_h, v'_h) \end{aligned} \tag{15}$$

$$\sum_{\bar{a}, \hat{a} = y}^{N_v} p_{\bar{a}} \geq 1 - \eta \tag{16}$$

Remarks

- Function (12) 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 (13) sets limit that traffic transmitted between any two hosts h_i and h_j must not exceed the corresponding bandwidth capacity $B(e_{ij})$.
- Constraints (14) and (15) put bounds that CPU and memory used by VMs coordinating with each other and by intra-host communications must not exceed C_h and M_h.
- Constraint (16) 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.

E. Scalability and Heuristic Algorithm

The formulation presented is one of the stochastic programming problems, which have been proved to be NP-hard [19]. 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 to achieve suboptimal results by applying a hybrid strategy of best-fit and first-fit decreasing algorithm.

As Algorithm 1 shows, when deploying a mobile edge application VM, all hosts are sorted as the first step: 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. Otherwise, the algorithm will try the host with the most remaining bandwidth to all other hosts with existing VMs deployed. 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 $O(N_v^2)$.

IV. Numerical Results

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

A. Assumptions and Parameters

To clearly demonstrate the trends we focus on, the following assumptions are made to simplify the modeling of the problem without losing the generality. Due to the limitation of paper length, we discuss the placement selection of the same mobile edge application with elasticity. This means variable number 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 include the same type of VMs with the same CPU, memory and network bandwidth requirements.

With the assumptions above, we choose some parameters in our model as constants, while others as variables, shown as Table I, 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

Algorithm 1: Availability-aware Mobile Edge Application Host Selection Algorithm

```
Data: A mobile edge application VM $v$
Data: All hosts $\mathbb{H}$
Result: Host $h$ to deploy $v$
begin
    host_list ← $empty$
    Add $h_1$ with most deployed VMs to host_list
    Sort other host by remaining bandwidth to all
      deployed VMs descending and add to host_list
    for $h \in$ host_list do
        $R_C(h) \leftarrow C_h$
        $R_M(h) \leftarrow M_h$
        foreach $v_h$ do
            $R_C(h) \leftarrow R_C(h) - C_{v_h} - \gamma_C(N_v - 1)$
            $R_M(h) \leftarrow R_M(h) - M_{v_h} - \gamma_M(N_v - 1)$
            if $v_h \neq v$ then
                $R_C(h) \leftarrow R_C(h) - \alpha_C B(v, v_h)$
                $R_M(h) \leftarrow R_M(h) - \alpha_M B(v, v_h)$
            end
        end
        if $R_C(h) < 0$ or $R_M(h) < 0$ then
            Skip $h$ and check the next host
        end
        foreach $v_j$ and $h \neq j$ do
            $R_B(e_{hj}) \leftarrow R_B(e_{hj}) - B(v, v_j)$
            if $R_B(e_{hj}) < 0$ then
                Skip $h$ and check the next host
            end
        end
        Calculate $p_y$ assuming $v$ on $h$
        if $p_y < 1 - \eta$ then
            Skip $h$ and check the next host
        end
        cost ← $S_C(h) + S_M(h) + S_B(e_{ij})$
        return $h$
    end
end
```

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.

B. 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, η is set to be 1 such that no minimum availability is required. This will help observe the trends of availability changes.

Figure 3 demonstrates the 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

TABLE I
PARAMETERS FOR AVAILABILITY-AWARE HOST SELECTION

Parameter	Value	Parameter	Value
C_v	2 vCPUs	M_v	2048 MB
α_C	10%	α_M	10%
M_h	102400 MB	C_h	100 vCPUs
$B(v, v')$	20 Mb	γ_C	10%
γ_M	0.1%	$B(e_{ij})$	10000 Mb
P_V	90%	P_H	90%
δ	0.1	y	*var*
W_C	*var*	W_M	*var*
W_B	*var*	η	*var*
N_v	*var*	N_h	*var*

see that increasing the number of backup VMs will 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, Figure 4 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.

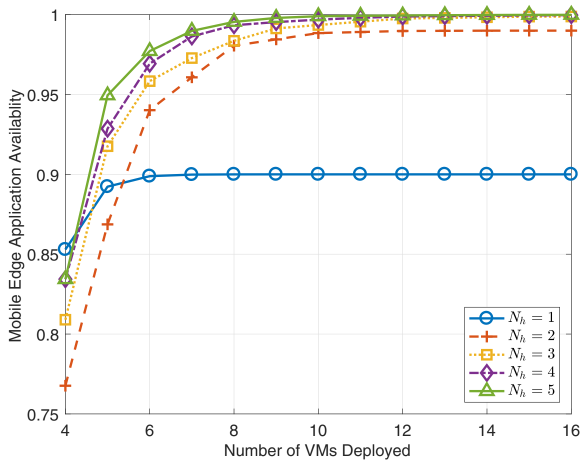


Fig. 3. **Availability of the mobile edge application with different numbers of VMs deployed and hosts.**

C. 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 $N_h = 10, N_v = 50, y = 2, W_C = W_M = 1$, and $\eta = 0.3\%$. We choose various value 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 Figure 5. 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

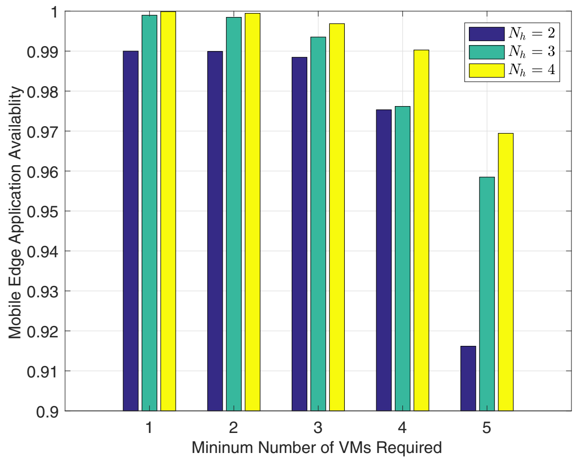
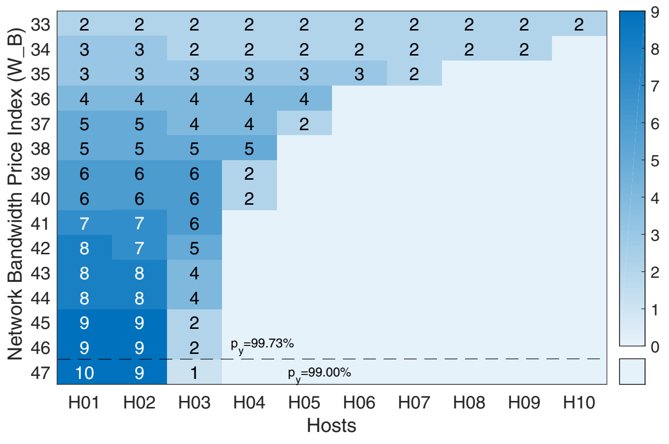


Fig. 4. **Availability of the mobile edge application with different numbers of VMs deployed and hosts.**

networks becomes congested, the algorithm would put VMs on less hosts to limited the 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 $W_B \geq 45$.



	H01	H02	H03	H04	H05	H06	H07	H08	H09	H10
33	2	2	2	2	2	2	2	2	2	2
34	3	3	2	2	2	2	2	2	2	
35	3	3	3	3	3	3	2			
36	4	4	4	4	4					
37	5	5	4	4	2					
38	5	5	5	5						
39	6	6	6	2						
40	6	6	6	2						
41	7	7	6							
42	8	7	5							
43	8	8	4							
44	8	8	4							
45	9	9	2							
46	9	9	2							
47	10	9	1							

Fig. 5. **Placement distribution of 20 VMs across 10 hosts with** $N_h = 10, N_v = 50, y = 2, W_C = W_M = 1,$ **and** $\eta = 0.3\%$. **Empty cell means no VM is deployed on the host with a certain** W_B.

V. Conclusion

In this paper, we have formulated a mobile edge application placement problem. Our model and proposed heuristic algorithm 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 is to consider mobile edge applications with combinations of different NFs and with service chaining.

References

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[2] 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.

[3] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile Edge Cloud System: Architectures, Challenges, and Approaches," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–14, 2017.

[4] S. E. Kevin Brown, Wendy Torell, "Edge computing needs reliability," http://www.datacenterdynamics.com/content-tracks/power-cooling/edge-computing-needs-reliability/97587.fullarticle, [Online; accessed Mar-24-2017].

[5] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer, 2014.

[6] M. Jammal, A. Kanso, and A. Shami, "High availability-aware optimization digest for applications deployment in cloud," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 6822–6828.

[7] S. Oechsner and A. Ripke, "Flexible support of VNF placement functions in OpenStack," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–6.

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

[9] ——, "Network Function Virtualisation(NFV); Architectural Framework," *ETSI GS NFV 002, V1.1.1 (2013-10)*.

[10] P. Veitch, M. J. McGrath, and V. Bayon, "An instrumentation and analytics framework for optimal and robust NFV deployment," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 126–133, 2015. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2015.7045400

[11] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges, AllThingsCellular 2014, Chicago, Illinois, USA, August 22, 2014*, 2014, pp. 33–38. [Online]. Available: http://doi.acm.org/10.1145/2627585.2627592

[12] Z. Usmani and S. Singh, "A survey of virtual machine placement techniques in a cloud data center," *Procedia Computer Science*, vol. 78, pp. 491 – 498, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916000958

[13] U. Bellur, C. S. Rao, and S. D. M. Kumar, "Optimal placement algorithms for virtual machines," *CoRR*, vol. abs/1011.5064, 2010. [Online]. Available: http://arxiv.org/abs/1011.5064

[14] A. Khosravi, S. K. Garg, and R. Buyya, "Energy and carbon-efficient placement of virtual machines in distributed cloud data centers," in *Euro-Par 2013 Parallel Processing - 19th International Conference, Aachen, Germany, August 26-30, 2013. Proceedings*, 2013, pp. 317–328.

[15] J. Zhu and C. Huang, "A universal protocol mechanism for network function virtualization and application-centric traffic steering," in *IEEE Conference on Standards for Communications and Networking, CSCN 2015, Tokyo, Japan, October 28-30, 2015*, 2015, pp. 257–262. [Online]. Available: http://dx.doi.org/10.1109/CSCN.2015.7390454

[16] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management, CNSM 2014 and Workshop, Rio de Janeiro, Brazil, November 17-21, 2014*, 2014, pp. 418–423. [Online]. Available: http://dx.doi.org/10.1109/CNSM.2014.7014205

[17] S. Lee, S. Pack, M.-K. Shin, and R. Browne, "Resource management in service chaining."

[18] A. Cassen, "Keepalived: Health checking for lvs & high availability," *URL http://www.linuxvirtualserver.org*, 2002.

[19] 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.