On Dynamic Hypervisor Placement in Virtualized Software Defined Networks (vSDNs)

Sen Chen, Weiqiang Sun, and Weisheng Hu
State Key Laboratory of Advanced Optical Communication Systems and Networks
Shanghai Jiao Tong University, Shanghai 200240, China
E-mail: sunwq@sjtu.edu.cn

ABSTRACT
Virtual software defined networking networks (vSDNs) combines the benefits of software defined networking (SDN) and network virtualization, and is important to make effective use of physical network resources. In vSDN environments, the control plane latency is the sum of the two parts: the latency from switch to its hypervisor and the latency from hypervisor to controller. Therefore, in order to optimize the latency of the network, the location of hypervisors need to be carefully decided. In this paper, we study the placement strategy of hypervisors when the location of hypervisor may be changed at run time. We mainly have two goals: minimize the latency of switch to its controller and minimize the impact caused by hypervisor migration. We define the problem as the dynamic hypervisor placement problem (DHPP) and formulate it into an Integer Linear Problem (ILP). Our evaluation results show that this minimizes the controller plane latency while incurring low hypervisor migration overhead.
Keywords: virtualized software defined networking networks, hypervisor placement, hypervisor migration.

1. INTRODUCTION

Software defined networking (SDN) is a networking paradigm that separates the control plane from the data plane which bring us open, programmable interfaces of the control plane [1]. Network virtualization (NV) allow multiple heterogeneous network architectures to cohabit on a shared physical substrate [2].By combining SDN and NV, virtual SDN networks (vSDNs) can be created on a given physical SDN network, and tenants can program their vSDN resources via open network interfaces and protocols [3]. Hypervisor plays an important role in virtualizing SDN networks. It abstracts the underlying SDN networks into multiple logically isolated virtual SDN networks (vSDNs), each with their own controller [4]. Hypervisor sits logically between tenant controller and switch (Fig. 1).

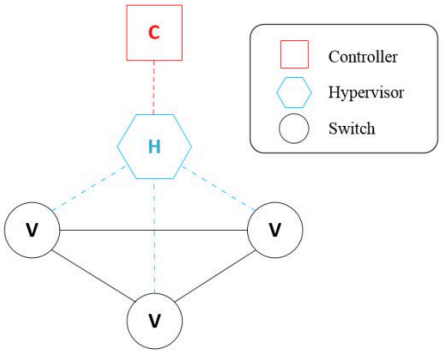


Figure 1. Hypervisor sits logically between tenant controller and switch.

In the non-virtualized SDN environment, switches are directly connected to the corresponding controller. In order to achieve better network performance such as average and worst case latency, controllers' quantity and location must be carefully decided. Because in virtualized SDN environments, open-flow message from switch to tenant controller must pass throw the corresponding hypervisor, the control plane latency is the sum of the latency from switch to its hypervisor and the latency from hypervisor to controller. Obviously, the newly introduced network element – hypervisor will increase the control plane latency, hence its placement has important implications on the control plane latency. The hypervisor placement problem (HPP) has been well studied in [3], [5] and [6], in which the authors assume that vSDN requests are known before hypervisors are placed. Thus, they model the HPP as a static facility location problem (FLP). However, in a large-scale WAN deployment, a static hypervisor configuration setup may not satisfy the dynamic adaption of the virtualization layer as vSDN requests may arrive over time. The optimal placement strategy for hypervisors will change as network conditions change. It means that hypervisors may migrate to other positions to dynamically adapt to demand fluctuation. We call this problem the Dynamic Hypervisor Placement Problem (DHPP). Different from the static vision of the problem, DHPP allow controller plane migration at run time to adjust to the network change. Even though hypervisor migrations can be achieved by using protocols such as the one proposed in [7],

it also will inevitably impact the operation of the network, which can be characterized by the migration time. Hence, the solution to the DHPP can be regarded as finding trade-offs between network performance (latency metrics) and migration overhead (time cost).

In this paper, we formulate DHPP as an Integer Linear Program (ILP) that considers both control plane latency and hypervisor migration time. We then carry out evaluations with our model in real network topologies. The results show that with the proposed model, we can strike a balance between latency and migration cost. The remainder of this paper is structured as follows. Section 2 introduces related work on controller placement, hypervisor placement and control plane migration. We give the system description in Section 3 and then formulate DHPP as ILP in Section 4. Evaluation results are shown in Section 5 and a brief summarize is in Section 6.

2. RELATE WORK

The controller placement problem in SDN was first discussed in [9]. Heller *et al.* wanted to figure out two questions: given a topology, how many controllers are needed and where to place them. They also analysed the trade-offs between the different latency metrics such as the worst-case and average latency. In [10], G Yao *et al.* took into account the load balance factor. They defined a capacitated controller placement problem (CCPP) and proposed an efficient algorithm to solve it. Hu *et al.* [11] and Müller *et al.* [12] considered the resilience and Survivability metrics. In [13], Bari *et al.* studied a dynamically vision of CPP, in which the number of active controller can be adjusted according to network changes. Blenk *et al.* [5] started the study of hypervisor placement problem (HPP). The main objective is to find the number of hypervisor instances needed and their placement. In addition to worst-case latency and average latency, they proposed other 2 metrics: average-maximum latency and maximum-average latency, and analyse the performance trade-offs when optimizing different latency metrics. Blenk *et al.* extended their work in [3]. They studied four different hypervisor architectures and introduced mixed integer programming formulations for these architectures. Killi *et al.* [6] rethink the controller and hypervisor placement problem, and proposed a strategy for determining the placement of controllers in a vSDN while fixing the hypervisor in the physical network (VCPP). They also proposed an approach for jointly optimizing the placement of hypervisors and controllers in a vSDN (JHCP). In order to satisfy the dynamics of virtual SDN networks, Basta *et al.* [7] proposed a control path migration protocol which allow a dynamic change of control connections between vSDNs and the tenants' controllers. He *et al.* [8] studied how controller migration and switch re-assignments affect the control plane performance. Their evaluation results reveal that relaxed migration times can bring big improvement over a non-dynamic SDN control plane.

3. SYSTEM DESCRIPTION

3.1 Network Models

We denote the substrate network by $G(N, E)$, where N is the set of substrate nodes and E is the set of substrate links. The substrate nodes $n \in N$ are connected by the undirected edges $e \in E$. We use $d_{s,t}$ to denote the time cost (latency) of the shortest path between two substrate nodes s and t.The potential hypervisor locations are given by the set H. (obviously, $H \subseteq N$). Let R denote the vSDN requests. A vSDN request $r \in R$ is composed of the set of vSDN nodes V_r and the vSDN controller c_r. We use $\pi(V_r)$ to denote a function that map the virtual nodes to the corresponding physical nodes.These notations are listed in Table 1.

3.2 Static Hypervisor Placement Problem (HPP)

For a given physical network G, the traditional hypervisor placement problem, which has been studied in [3], [5] and [6], assumed that all vSDN requests are known in advance, i.e., the set of vSDN requests R, is taken as input. In order to minimize the latency metrics, e.g., the worst case latency or the average latency, we need to choose the locations of k hypervisors among all potential hypervisor locations H and decide which hypervisor these switches should be assigned to.

Table 1. Notations.

Symbol	Description
$G(N,E)$	Substrate network
N	Set of substrate nodes
E	Set of substrate links
H	Set of potential hypervisor locations
$d_{s,t}$	Latency of shortest path between substrate nodes s and t
R	Set of vSDN requests
r	vSDN request $r \in R$
V_r	Set of virtual nodes of vSDN request r
c_r	Virtual controller node of vSDN request r
$\pi(V_r)$	Mapping from virtual node to corresponding physical node

3.3 The Dynamic Hypervisor Placement Problem (DHPP)

Traditional Hypervisor Placement Problem is a static optimization problem, with fixed hypervisors position after get the solution of the problem. However, as vSDN requests are added dynamically in an SDN based public cloud environment, runtime updates of hypervisor instances should be taken into consideration. In other words, we should develop a new model to optimize the placement of hypervisors as virtual network demands change at runtime. This is what we call the Dynamic Hypervisor Placement Problem (DHPP). In DHPP, besides the latency metrics, we also consider the overhead caused by migration of hypervisors. That is, we want to minimize the control plane latency while keeping hypervisor migration cost low.

4. PROBLEM FORMULATION

In this section, we formulate DHPP as an integer linear programming (ILP) problem.

4.1 Decision Variables

The variable y_j is set to one if a hypervisor instance is deployed at potential hypervisor location $j \in H$, otherwise it's zero. The variable $x_{i,j}$ indicates whether the physical node $i \in N$ is controlled by the hypervisor node $j \in H$. The variable $\widetilde{x_{i,j}}$ indicates whether the physical node $i \in N$ is controlled by the hypervisor node $j \in H$ in the last optimization result. The variable $t_{n,m}$ is set to one if $n \neq m$ .These binary decision variables are listed in Table 2.

Table 2. Binary decision variables.

Notation	Description
y_i	=1, if a hypervisor is deployed at potential hypervisor location $j \in H$; 0, otherwise
$x_{i,j}$	=1, if the physical node $i \in N$ is controlled by the hypervisor node $j \in H$; 0, otherwise
$\widetilde{x_{i,j}}$	=1, if the physical node $i \in N$ is controlled by the hypervisor node $j \in H$ in the last optimization result; 0, otherwise
$t_{n,m}$	=1, if $n \neq m$; 0, otherwise

4.2 The Optimization Model

The goal of DHPP is to get a trade-off of two sides: minimize the latency of switch to its controller and minimize the impact due to the hypervisor migration. For the latency part, we mainly care about two kinds of latency metrics: maximum latency L_{max} (worst case latency) and average latency L_{ave}. They can be calculated in (1) and (2):

$$L_{max} = max_{r \in R,\ i \in V_r,\ j \in H}\ x_{i,j}\left(d_{\pi(i),j} + d_{j,\pi(c_r)}\right) \tag{1}$$

$$L_{ave} = \frac{1}{\sum_{r \in R} |V_r|} \sum_{r \in R} \sum_{i \in V_r} \sum_{j \in H} x_{i,j}\left(d_{\pi(i),j} + d_{j,\pi(c_r)}\right) \tag{2}$$

To minimize the impact of hypervisor migration, we must figure out which kind of hypervisor management protocol are used. Here, we refer to the work of Basta *et al.* [7]. The authors propose an effective control path migration protocol to support the dynamic changes of control connections. According to their protocol, if switch i is connected to the controller j towards the initial hypervisor m, and we plan to make a hypervisor migration from m to a new hypervisor n, 11 times handshake of i to n, 5 times handshake of n to j and 1 time handshake of i to n are needed. We use T to denote the time cost in a hypervisor migration:

$$T = C_1 d_{i,n} + C_2 d_{n,j} + C_3 d_{i,m} \tag{3}$$

Here, $C_1 = 11. C_2 = 5, C_3 = 1$.Obviously, it is possible that more than one hypervisor will migrate, so we use T_{max} to denote the maximum value of all possible T. The objective of our optimization model is to minimize the weighted sum of latency metrics and migration time cost:

$$z = \alpha L + \beta T_{max} \tag{4}$$

Here, α and β are adjustable constants, which can be set according to the relative significance of the two parts. Now the DHPP can be formulated as follows:

$$min\ z \tag{5}$$

Subject to the following constraints:

$$\sum_{j \in H} x_{i,j} = 1 \qquad \forall r \in R,\ i \in V_r \tag{6}$$

$$\sum_{i \in H} y_i = h \tag{7}$$

$$y_j \geq x_{i,j} \qquad \forall r \in R,\ i \in V_r,\ j \in H \tag{8}$$

$$\sum_{i \in V_r} x_{i,j} \geq y_j \qquad \forall r \in R, j \in H \tag{9}$$

$$T_{max} \geq x_{i,n}\widetilde{x_{i,m}}t_{n,m}\left(C_1 d_{i,n} + C_2 d_{n,c_r} + C_3 d_{i,m}\right) \qquad \forall r \in R, i \in V_r, n \in H, m \in H \tag{10}$$

$$y_j \in \{0,1\} \qquad \forall j \in H \tag{11}$$

$$x_{i,j} \in \{0,1\} \qquad \forall i \in V_r, r \in R, j \in H \tag{12}$$

$$\widetilde{x_{i,j}} \in \{0,1\} \qquad \forall i \in V_r, r \in R, j \in H \tag{13}$$

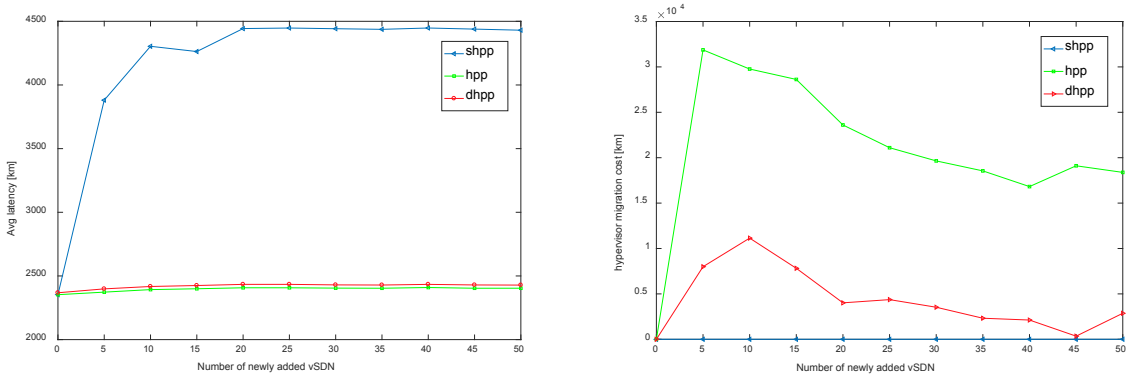
$$t_{n,m} \in \{0,1\} \qquad \forall i \in V_r, r \in R, j \in H \tag{14}$$

Equation (6) ensures that every switch is assigned to exactly one hypervisor to connect to its controller. Equation (7) guarantees the total number of hypervisors is exactly equal to h. Constraint (8) ensures that a hypervisor is installed at location j if a switch i is connected to its controller via the hypervisor j. Constraint (9) ensures that if no switch connects to its controller via the hypervisor j, then there is no hypervisor installed on j. Constraint (10) guarantees that, for all possible hypervisor migrations, the maximum migration time T_{max} is greater than or equal to its migration time cost. Constraints (11)-(14) forces all decision variables to take values zero or one.

5. EVALUATION RESULTS

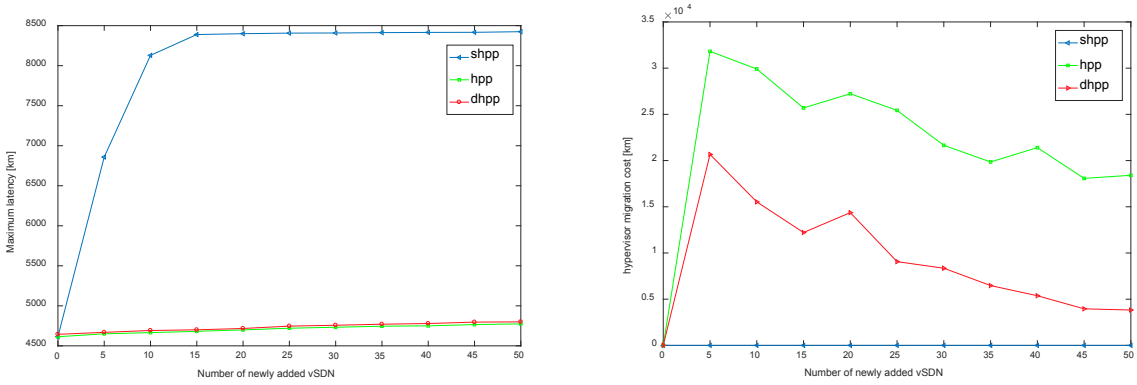
We use AT&T North America network (25 nodes and 57 links) as the evaluation topology. The number of vSDN request nodes is randomly distributed between 2 and 10. We assume that each virtual network is controlled by a single controller which is deployed at a random position among all possible location set H. The DHPP, as an Integer Linear Problem, is modelled by C++ in Microsoft Visual Studio and solved by IBM CPLEX Optimizer [14]. In ordered to analyse the performance of DHPP, we also run the traditional HPP algorithm and the static HPP algorithm (SHPP) simultaneously. We first use 20 vSDN requests to determine the initial placement of hypervisors. Obviously, in this stage, the performance of DHPP, HPP and SHPP is completely consistent as there is no hypervisor migration occurred. Then the vSDN requests are being added dynamically. For the traditional HPP algorithm, the cost of hypervisor migration is not considered, it just calculate the optimal solution of the latency metrics. In the static HPP algorithm, the placement of hypervisor remains the same, it just needs to consider how to assign these hypervisors to the newly added vSDN nodes. DHPP algorithm make a balance of latency metrics and hypervisor migration overhead. All our simulations were performed on a single machine with Intel Xeon E3-1231 processor at 3.4GHZ with 16GB of memory. To improve accuracy of evaluation results, each instance is executed for 100 times and take the average.

Figure 2 illustrate the performance of SHPP, HPP and DHPP when they are optimized for average latency. Figure 2(a) shows the average latency of three algorithms as the number of vSDN requests increases. Obviously, SHPP gets the worst latency as it can't adjust the placement of hypervisor for newly added vSDN requests. DHPP performs just a litter bit worse than HPP. Figure 2(b) shows the hypervisor migration cost. SHPP's value is always zero as no migration exists in this case. With HPP and DHPP, the migration cost will increase first then decreasing slowly. This is because as the number of requests increases, the placement of hypervisor becomes fixed gradually. We can see that DHPP performs much better than HPP in hypervisor migration cost while they have almost same average latency. The performance when these algorithms are optimized for maximum latency is illustrated in Fig.3, with almost the same result compared to the average latency.



(a) average latency (b) hypervisor migration cost

Figure 2. Performance of SHPP, HPP and DHPP while optimized for the average latency.



(a) maximum latency (b) hypervisor migration cost

Figure 3. Performance of SHPP, HPP and DHPP while optimized for the maximum latency.

6. CONCLUSIONS

In this paper, we are interested in optimizing control delay in vSDNs when hypervisors can be relocated. We define this problem as the Dynamic Hypervisor Placement Problem (DHPP). DHPP takes into account the control plane latency and hypervisor migration cost. We provided an ILP formulation for the DHPP, then evaluate the performance of DHPP in AT&T NA network. Our results shows that DHPP achieve a good trade-off between latency metrics and hypervisor migration overhead.

ACKNOWLEDGEMENTS

We acknowledge the support of the National Natural Science Foundation of China (61901118 and 61433009).

REFERENCES

[1] D. Kreutz *et al*.: Software-defined networking: A comprehensive survey, *arXiv* preprint:1406.0440 (2014).
[2] N. M. Chowdhury, M. Kabir, and R. Boutaba: Network virtualization: State of the art and research challenges, *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20-26, 2009.
[3] A. Blenk *et al*.: Control plane latency with SDN network hypervisors: The cost of virtualization, *IEEE Transactions on Network and Service Management*, vol. 13, no. 3. pp. 366-380, 2016.
[4] A. Blenk *et al*.: Survey on network virtualization hypervisors for software defined networking, *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 655-685, 2015.
[5] A. Blenk *et al*.: Pairing SDN with network virtualization: The network hypervisor placement problem, in *Proc. 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015.
[6] B. P. R. Killi and S. V. Rao: On placement of hypervisors and controllers in virtualized software defined network*, IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 840-853, 2018.
[7] A. Basta *et al*.: Towards a dynamic SDN virtualization layer: Control path migration protocol, in *Proc. 2015 11th International Conference on Network and Service Management (CNSM)*, 2015.
[8] Mu He *et al*.: How flexible is dynamic SDN control plane?, in *Proc. 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017.
[9] B. Heller, R. Sherwood, and N. McKeown: The controller placement problem, in *Proc. First Workshop on Hot Topics in Software Defined Networks, ACM*, 2012.
[10] Guang Yao *et al*.: On the capacitated controller placement problem in software defined networks, *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339-1342, 2014.
[11] Yannan Hu *et al*.: On reliability-optimized controller placement for software-defined networks, *China Communications*, vol. 11, no. 2, pp. 38-54, 2014.
[12] L. F. Müller *et al*.: Survivor: An enhanced controller placement strategy for improving SDN survivability, in *Proc. 2014 IEEE Global Communications Conference*, 2014.
[13] M. F. Bari *et al*.: Dynamic controller provisioning in software defined networks, in *Proc. CNSM*, 2013.
[14] IBM CPLEX Optimizer. https:// https://www.ibm.com/analytics/cplex-optimizer.