Pairing SDN with Network Virtualization: The Network Hypervisor Placement Problem

Andreas Blenk, Arsany Basta, Johannes Zerwas, Wolfgang Kellerer
Chair of Communication Networks
Department of Electrical and Computer Engineering
Technische Universität München, Germany
Email: {andreas.blenk,arsany.basta,johannes.zerwas,wolfgang.kellerer}@tum.de

***Abstract*—A network virtualization hypervisor for Software Defined Networking (SDN) is the essential component for the realization of virtual SDN networks (vSDNs). Virtualizing software defined networks enables tenants to bring their own SDN controllers in order to individually program the network control of their virtual SDN networks. A hypervisor acts as an intermediate layer between the tenant SDN controllers and their respective virtual SDN networks. The hypervisor consists of the network functions that are necessary for virtualization, e.g., translation or isolation functions. For scalability, the hypervisor can be realized via multiple physically distributed instances each hosting the needed virtualization functions. In this way, the physical locations of the instances, which realize the hypervisor, may impact the overall performance of the virtual SDN networks. Network virtualization adds new dimensions to the general SDN controller placement problem. This paper initiates the study of the network hypervisor placement problem (HPP). The HPP targets the following questions: How many hypervisor instances are needed? Where should the hypervisor instances be placed in the network? For our study of the HPP, we provide a mathematical model that solves the HPP for a case where node and link capacity constraints are not considered. We propose four latency metrics for optimizing placement solutions based on our model for vSDNs. Covering a real network topology, our evaluation quantifies the trade-offs between the new metrics when used as objectives. Furthermore, we analyze the impact of the physical network topology on the optimization results and identify potentials for improvement, e.g., in terms of runtime.**

I. Introduction

Network virtualization (NV) is seen as a key enabler to improve the resource efficiency of today's communication networks by sharing network resources between multiple tenants [1]. Software Defined Networking (SDN) introduces flexible and dynamic programming of networks at run time [2]. Combining NV and SDN provides the benefits of both concepts. Receiving their own virtual SDN network (vSDN), tenants can bring their own controllers in order to run their individual network operating systems that are specifically designed for their networking demands. For instance, the programmability of slices (virtual networks) is an integral part of the future 5G network architecture [3]. Further, for the Network Functions Virtualization (NFV) use case, virtual SDN networks provide the means to dynamically interconnect virtual network functions flexibly at runtime.

In order to virtualize SDN networks, recent work has proposed solutions that introduce a network virtualization layer [4], [5], or network hypervisor [1]. The network hypervisor, or shortly hypervisor, consists of one or multiple instances that are hosting the virtualization functions, e.g., isolation functions, translation functions, or abstraction functions. For example, isolation functions monitor and isolate the control traffic between tenant controllers. This is needed to prevent one tenant controller to overload a switch, thus impacting the network performance of other tenants. Accordingly the hypervisor sits logically between the physical SDN networks, where the vSDNs are hosted, and the tenant controllers, which may run an individually designed network operating system for each vSDN.

In non-virtualized SDN networks, the SDN controller performance and the design of the control plane can have a significant impact on the network performance [6], [7]. In particular, a low flow setup latency is important for the SDN network performance. High control plane latencies can lead to state inconsistencies of applications that are running on top of SDN controllers [8]. Accordingly, when deploying SDN networks, the distribution of the control plane should be taken into account. Besides the impact of the implementation, the placement of the decision logic for non-virtualized SDN networks, i.e., the SDN controllers, can significantly impact the decision latency [9]. A large amount of work has already been proposed to solve the controller placement problem efficiently [9], [10], [11], [12], [13]. The existing research optimizes the controller placement for different metrics, e.g., control plane latency and resilience, and for different network types, e.g., wide area networks (WANs). As multiple virtual SDN networks are co-existing in a virtual SDN environment, the demands and needs of each tenant for SDN network performance can directly be applied to the network hypervisor placement problem.

As already mentioned, in case of virtualized SDN networks, the hypervisor processes all network control traffic exchanged between tenant controllers and their virtual SDN networks. In particular for long-propagation-delay WANs, the hypervisor instances have to be distributed for scalability reasons. As for non-virtualized SDN networks, an unplanned hypervisor placement may lead state inconsistencies of the applications of the tenants [8]. The locations of the distributed hypervisor instances can affect the control plane latency of the individual vSDNs. In case of the SDN controller placement

[1]Note that we use the term hypervisor for simplicity. However, this term should not be confused with server virtualization, which is not addressed in this paper.

problem, it has been shown that not every physical SDN node has to be served by one controller. This means that a minor number of SDN controller suffices to serve specific use cases [9]. Similar to the SDN controller placement problem, it has accordingly to be determined how many hypervisor instances are needed and where to place them inside the network. Hence, the decision where to place the hypervisor(s) has to be seriously considered in order to provide the best possible performance for vSDNs.

Our contributions are to initiate the study of the network hypervisor placement problem (HPP) for virtualized SDN networks. We provide a first mathematical model for the HPP. Given an SDN network topology with possible hypervisor locations and a number of vSDNs, the model can be used to determine the number of hypervisor instances and their locations. Unlike the initial SDN controller placement problem, we propose four latency performance metrics. Two metrics are proposed to quantify the performance per virtual SDN network. Via simulations for a real network topology, we quantify the trade-offs between the four optimization metrics.

The remainder of this paper is structured as follows. In Section II, we introduce and illustrate the HPP. In Section III, we identify and summarize related work. In Section IV, we outline our models to solve HPP. Section V presents our simulation results for a real network topology. Finally, we draw conclusions in Section VI.

II. Network Hypervisor Placement Problem

The network hypervisor placement problem (HPP) deals with two fundamental questions:

1) *How many hypervisor instances are needed?*
2) *Where should the hypervisor instances be placed?*

Figure 1 illustrates the HPP for one vSDN network. The tenant controller location and the locations of the vSDN network elements are given. The blue solid line illustrates the connection between the vSDN controller **(C)** and the hypervisor **(H)**. The green dashed lines show the connections between the hypervisor and the corresponding vSDN nodes **(V)**. Note that not all physical SDN nodes are used by the vSDN tenant. Accordingly, only the populated physical SDN nodes and the tenant location have to be taken into account for the HPP. During vSDN network operation, control traffic is only transmitted via the physical paths that connect the populated nodes with the hypervisor. The control plane latency between a tenant controller and one vSDN node is the sum of the latency of the path connecting the vSDN node with the hypervisor and the latency of the path connecting the hypervisor with the controller.

III. Related Work

In the context of SDN virtualization, there are several research areas which are related to the HPP. We first introduce the classical controller placement problem and then analyze extensions with respect to different metrics. We further shortly outline how the general facility location problem, virtual network embedding, and network function virtualization are related to the hypervisor placement problem.

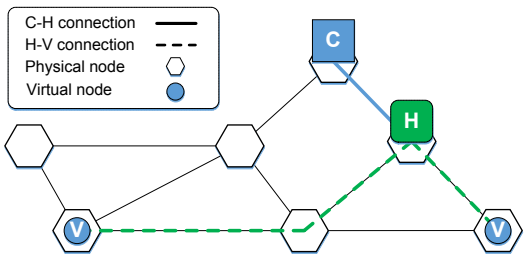


Fig. 1. Exemplary hypervisor placement. Hexagons show the physical network locations. Small dashed lines indicate the physical connections. Controller (C) connects via hypervisor (H) to vSDN nodes (V)

SDN Controller Placement Problem In general, the design of the SDN control plane leads to the questions of how many controllers are required and where to place them. This controller placement problem (CPP) has been initiated by Heller et al. [9]. In the following, we outline related work to the CPP with focus on optimization metrics, which are load, latency, and resilience.

Load and Latency. Heller et al. [9] initiated the study on the CPP with a proposed brute force solution. They introduce the average and the maximum (worst case) latency as evaluation metrics for the CPP solutions. They evaluate the impact of different placements on the latency metrics for real network topologies. Yao et al. [13] extend the problem by considering limited capacity of network elements. They show that considering limited capacities is needed to avoid an overload of network elements. [14] provides a detailed mathematical model of the controller placement. The model allows to simultaneously determine the number, the type, and the location of the controllers. A game theory-based scheme is provided that runs distributed on each controller instance in [12]. Each node that can act as a controller decides whether an additional controller is required. The scheme can ensure the optimal number of controllers and the appropriate locations. Xiao et al. [15] provide a method to deal with SDN in WANs. The network is divided into several sub-domains by the use of spectral clustering. The controller placement is then solved for each of these domains. To measure the performance in the domains, metrics for the spectral clustering are introduced. In their evaluation, the authors compare spectral clustering algorithm to optimizing for average latency.

Resilience. Multiple different concepts exist that extended the classical CPP with respect to resilience [16], [17], [18], [19], [20], [11]. In general, they mostly analyze the trade-offs between the classical CPP metrics and resilience [20]. However, our initial study does not consider resilience for the hypervisor placement.

Facility Location Problem. From a mathematical point of view the Facility Location Problem (FLP) is similar to the CPP. In [21] an overview of different types of the FLP and several models is given. Pirkul and Jayamaran [22] provide a mixed integer program for a multi-commodity capacitated version of the FLP. They propose an LP relaxation and a heuristic algorithm, which reduce the solution time for larger problems. Campbell [23] describes several types of the discrete version of the FLP and provides mathematical formulations of

the problems. The proposed models are adapted and applied to the model for the hypervisor placement problem in this paper.

Virtual Network Embedding. Regarding the embedding of virtual SDN networks, Fischer et al. [24] give an overview and classify different Virtual Network Embedding (VNE) algorithms. They present metrics for the VNE and create a taxonomy for the algorithms. For the embedding of virtual SDN networks, [25] proposes different algorithms (greedy, simulated annealing). Both algorithms can optimize for either a latency metric or load-balancing. The authors show that there is a trade-off between the metrics. Additionally the impact of the embedding of the controller node on the performance is shown. However, the performance study does not consider the placement of the hypervisor instances.

Network Function Virtualization. Luizelli et al. [26] propose a formulation for the embedding of virtual network function chains on a network infrastructure. The proposed model targets a minimum number of virtual network functions to be mapped on the substrate. Their evaluation metrics mainly considered infrastructure resource utilization, e.g., CPU, as well as end-to-end latency of the embedded function chains. However, the proposed model does not incorporate the characteristics of SDN networks, moreover virtual SDN networks.

IV. HPP Formulation

In this section, a generic Mixed Integer Linear Program (MILP) formulation for the HPP is introduced where limiting capacities of physical nodes or edges are not considered. The problem is formulated as a path-flow, undirected, and non-splittable model. Four different objectives are evaluated as variations to the model. The objectives target several measures for the latency between virtual SDN networks and their respective controllers.

A. Sets and Parameters

First, the problem definition is described mathematically. The virtual SDN network embedding and the controller placement are assumed to be given. The problem solution focuses on the placement of the hypervisor(s), i.e., hypervisor instances. An overview of the problems' sets and parameters is presented in Table I.

1) Substrate: We are considering a substrate network $G = (V, E)$. The set V represents the physical nodes, which correspond to locations. The physical nodes are connected by a set of undirected edges E. At each physical node (location) $v \in V$, there can be none, one or multiple virtual nodes embedded.

2) Hypervisor Locations: $H \subseteq V$ is the set of possible hypervisor locations, whereby the hypervisors can be co-located with the physical SDN nodes. In the simplest case H, equals V. However, a selection of candidate locations for the hypervisor can be done as well. The number of hypervisors that need to be set is denoted by the constant Q.

3) Virtual SDN Networks: The set M represents the virtual networks. On the given substrate network, $|M|$ virtual networks are embedded. Each virtual network $m \in M$ has $|N_m|$ virtual SDN nodes, with $N_m \subseteq V$. The physical node $c_m \in V$ defines the location of the SDN controller of the virtual network m.

TABLE I. SETS AND PARAMETERS

Substrate Network	$G = (V, E)$
Physical Node	$w \in V$
Possible Hypervisor Node	$v \in H \subseteq V$
Number of Hypervisors	Q
Virtual Network	$m \in M$
Virtual Node of Virtual Network m	$n \in N_m \subseteq V$
Virtual Controller of Virtual Network m	$c_m \in V$
Demand of Virtual Network m	$d \in D_m$
Cost for Path $d \in D_m$ via $v \in H$	$\phi_{m,d,v}$

4) Demands and Paths: Each virtual node of a virtual network $m \in M$ needs a control link to the corresponding controller. Since the control link has to pass through the SDN hypervisor, there are $|N_m|$ control link demands for each virtual network. The set of the demands is D_m. A demand $d = (n, c_m) \in D_m$ is specified between a virtual SDN node n and the corresponding controller c_m. For each demand d and a potential hypervisor node, the shortest path is computed between the virtual node and its controller, which traverses the potential hypervisor node. The cost for the path of demand d via hypervisor node v is the summed cost $\phi_{m,d,v}$ of the traversed links. A path where source node, intermediate node and destination node have the same location, has cost 0.

B. Variables

For each possible hypervisor location there is a binary variable x_v that indicates if a hypervisor instance is placed on the physical node $v \in H$. For each demand d and possible hypervisor node v there is a binary selection variable $y_{m,d,v}$. If $y_{m,d,v} = 1$, then the path of demand d via hypervisor instance at v is selected. The binary variable $z_{w,v}$ indicates if physical node $w \in V$ is controlled by the hypervisor instance $v \in H$.

C. Constraints

In the following the constraints are imposed.

1) Hypervisor installation: Constraint (1) ensures that the number of placed hypervisor instances is Q:

$$\sum_{v \in H} x_v = Q. \tag{1}$$

2) Path selection: For each demand of all virtual SDN networks on the substrate, exactly one path has to be selected. This is ensured by Constraint (2):

$$\sum_{v \in H} y_{m,d,v} = 1 \quad \forall m \in M,\ \forall d \in D_m. \tag{2}$$

Constraint (3) sets a hypervisor node v to be chosen in case a path that includes this hypervisor node is selected. The sum over all demands of all virtual network is intended to reduce the total constraints' space. It addresses the two boundaries, where no demands from all virtual networks select a hypervisor node v as well as all demands from all virtual networks decide for this hypervisor node. At least one demand in all virtual networks is required to set the selection of the hypervisor node.

$$\sum_{m \in M} \sum_{d \in D_m} y_{m,d,v} \leq \left(\sum_{m \in M} |D_m| \right) x_v \quad \forall v \in H. \tag{3}$$

3) Physical Node Assignment: We assume that a physical SDN node is controlled by only one hypervisor instance. This assumption aims at avoiding control conflicts or state inconsistencies that may occur if an SDN switch has multiple controllers. Constraint (4) sets a hypervisor node v to be the controller of the physical SDN node w where the virtual node n that belongs to demand d is embedded, i.e., $w = n(d)$:

$$y_{m,d,v} \leq z_{w,v} \quad \forall m \in M,\ \forall d \in D_m,\ \forall w \in V,\ \forall v \in H. \quad (4)$$

Constraint (5) ensures that each physical SDN node is controlled by a single hypervisor instance:

$$\sum_{v \in H} z_{w,v} \leq 1 \quad \forall w \in V. \quad (5)$$

D. Latency Metrics/Objectives

Average and maximum latency are often used metrics in the area of controller placement. Both are extended to the HPP. Furthermore, two additional metrics are provided. Appropriate constraints and objectives are presented if needed to optimize for a particular metric:

1) Maximum (Worst Case) Latency: This metric targets the worst case latency, i.e., minimize the latency of the demand that has the longest path among all vSDNs:

$$L_{max} = \max_{m \in M, d \in D_m} \sum_{v \in H} y_{m,d,v} \phi_{m,d,v}. \quad (6)$$

Equation (6) can be formulated in the model by adding a constraint to represent the maximum latency as a continuous variable w:

$$\sum_{v \in H} y_{m,d,v} \phi_{m,d,v} \leq w \quad \forall m \in M,\ \forall d \in D_m. \quad (7)$$

The actual objective is to minimze the maximum latency:

$$\min w. \quad (8)$$

2) Average Latency: This metric targets a minimum average latency over all demands from all virtual networks:

$$L_{avg} = \frac{1}{\sum_{m \in M} |D_m|} \sum_{m \in M} \sum_{d \in D_m} \sum_{v \in H} y_{m,d,v} \phi_{m,d,v}. \quad (9)$$

The corresponding objective is:

$$\min L_{avg}, \quad (10)$$

whereby no additional constraints need to be defined.

3) Average-Maximum Latency: This metric targets the worst case latency per vSDN. It considers the maximum latency of each vSDN and aims at a minimum average of all considered maximum values:

$$L_{avg,max} = \frac{1}{|M|} \sum_{m \in M} \max_{d \in D_m} \sum_{v \in H} y_{m,d,v} \phi_{m,d,v}. \quad (11)$$

To optimize for this latency, $|M|$ new continuous variables w_m representing the maximum latency of a virtual network m are required. The variables w_m are set to the maximum latency of vSDN m by the constraint:

$$\sum_{v \in H} y_{m,d,v} \phi_{m,d,v} \leq w_m \quad \forall m \in M,\ \forall d \in D_m. \quad (12)$$

TABLE II. EVALUATION SETTINGS

Parameter	Values
no. of vSDNs $\|M\|$	$1, 3, 5, 7, 10, 15, 20, 40, 70, 100, 140$
no. of virtual nodes per vSDN $\|N_m\|$	uniformly distributed $2 \ldots 10$
controller location for each vSDN c_m	uniformly selected from $v \in V$
virtual node location $n_i \in N_m$	uniformly selected from $v \in V$
runs per model	200
HPP objectives	Max, Avg, AvgMax, MaxAvg

The actual objective is:

$$\min \frac{1}{|M|} \sum_{m \in M} w_m. \quad (13)$$

4) Maximum-Average Latency: This metric targets the average latency per vSDN. It considers the average latency of each vSDN and aims at a minimum for the maximum of all considered averages:

$$L_{max,avg} = \max_{m \in M} \frac{1}{|D_m|} \sum_{d \in D_m} \sum_{v \in H} y_{m,d,v} \phi_{m,d,v}. \quad (14)$$

Similar to Section IV-D1, a continuous variable w representing the maximum average latency of all virtual networks is required:

$$\frac{1}{|D_m|} \sum_{d \in D_m} \sum_{v \in H} y_{m,d,v} \phi_{m,d,v} \leq w \quad \forall m \in M. \quad (15)$$

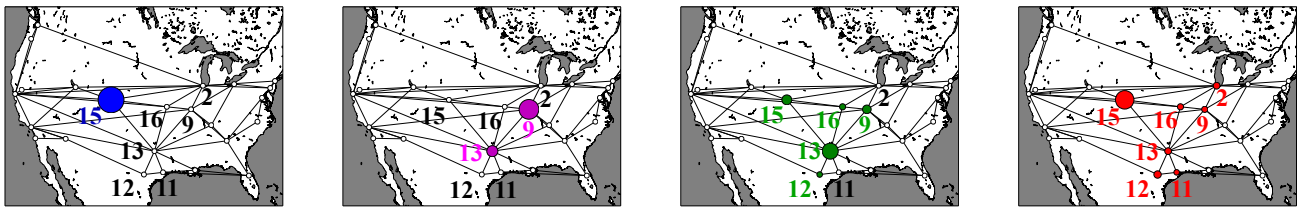
The actual objective is:

$$\min w. \quad (16)$$

V. ANALYSIS OF ATT NORTH AMERICA FOR SINGLE HYPERVISOR

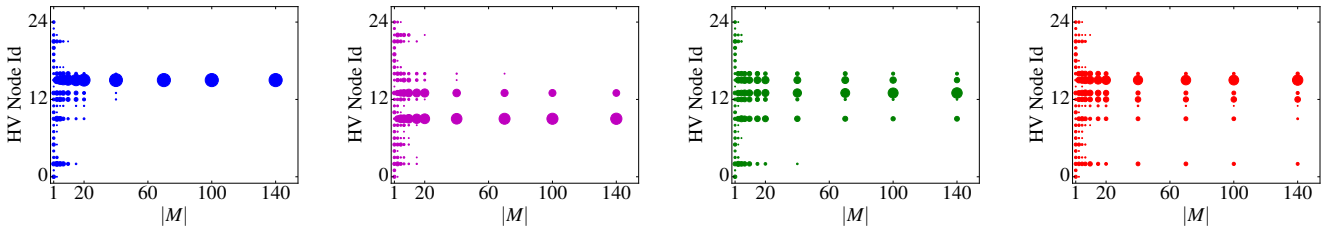
We investigate the trade-offs between all metrics when they are applied as an objective for placing a single hypervisor instance ($Q = 1$). The chosen network is the ATT North America topology [27], which consists of 25 nodes. We demonstrate the impact of different evaluation setups on the outcome of all four optimization objectives.

A python-based framework using Gurobi [28] was implemented for all four models. The parameter settings are given in Table II. As explained in Section II, not all physical nodes have to be used by a vSDN tenant. Accordingly, we evaluate the HPP for different numbers $|M|$ of vSDN networks. We call this the *vSDN network density*. For example, for $|M| = 1$ only one virtual SDN network is considered during the hypervisor placement while $|M| = 140$ means that 140 virtual SDN networks are considered. For all vSDN networks $m \in M$, we assume that the virtual nodes and their controllers are given as an input. The number of virtual nodes $|N_m|$ per vSDN m is uniformly distributed between 2 and 10. The vSDN node locations are uniformly selected from all available physical network locations V. For each vSDN, the controller location c_m is also uniformly selected from all physical network locations V. Each setting is repeated 200 times for varying vSDN networks for statistical evidence.



(a) Min. maximum latency (b) Min. average latency (c) Min. average-maximum latency (d) Min. maximum-average latency

Fig. 2. Hypervisor placement considering the four objective metrics ($Q = 1, |M| = 140$).



(a) Min. maximum latency (b) Min. average latency (c) Min. average-maximum latency (d) Min. maximum-average latency

Fig. 3. Hypervisor placement considering the four objective metrics for increasing number of vSDNs ($Q = 1, |M| = [1, 140]$).

A. Is there a dominant location for the network hypervisor under different objectives?

We start by comparing the different objectives with respect to the geographical locations of the hypervisor nodes. The four Figures 2(a)-(d) show the resolved SDN hypervisor locations for each objective metric in case $|M| = 140$. The size of a circle of a node represents the frequency of how often the node is selected as the optimal hypervisor location over 200 runs. A physical node location that is colored white with the smallest possible size indicates that the node is never selected for the hypervisor location.

Figure 2(a) depicts the results for minimizing the maximum latency. The hypervisor location converges to the network node 15, which lies in the sparse part of the network. As the maximum latency is mostly impacted by long switch to controller links, the hypervisor location is accordingly biased with the longer links connecting between the east and west parts of the network. Figure 2(b) shows two dominant hypervisor locations towards the center of the network (node 9 and 13). In contrast to solving the model for the maximum latency, minimizing the average latency considers all controller to virtual node connections of all vSDNs. As the vSDNs are randomly distributed, the hypervisor locations converge to the more populated part of the network, i.e., the east of the network. Figure 2(c) illustrates the hypervisor locations with a minimum average-maximum latency. As this model considers the average of the maximum latency of all vSDNs, more physical locations (node 9, 12, 13, 15, 16) for the hypervisor are optimal among the 200 runs. It can be observed that the hypervisor locations are overlapping with the locations as shown in Figure 2(b). Finally, minimizing the maximum-average latency, shown in Figure 2(d), considers the maximum of the average latency within each vSDN. This model results again in more candidate locations for the hypervisor (node 2, 9, 11, 12, 13, 15, 16). The dominant hypervisor location is the same as in Figure 2(a).

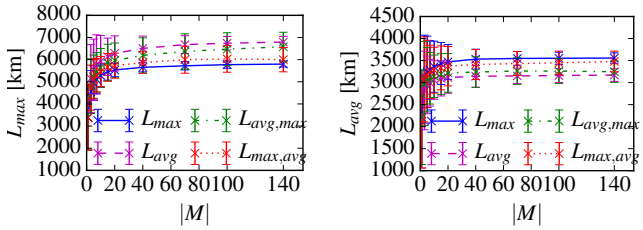
To conclude, we can observe that only a subset of nodes are resolved as candidate hypervisor locations under the given input parameters. Besides, the candidate hypervisor locations are neighboring. When average-maximum and maximum-average are considered as objectives, more nodes are providing an optimal solution for the hypervisor location.

B. How does the vSDN network density influence the hypervisor placement?

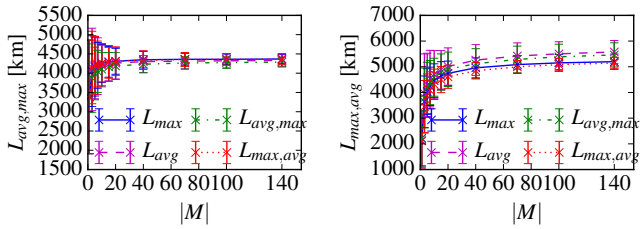
Figure 3 shows the impact of the vSDN density, i.e., number $|M|$ of vSDNs. While the x-axis shows the vSDN density, the y-axis shows the numbers of the network nodes. Again, the size of the circles indicate how often a location is chosen among 200 runs.

In case of a single vSDN network ($|M| = 1$), all physical nodes are selected at least one time as a hypervisor location within the 200 runs. This can be explained by the fact that just one vSDN network with 7 nodes on average is placed randomly among the network. This single vSDN determines the hypervisor location for all metrics. Accordingly, more potential hypervisor locations are optimal for each individual run. With an increasing vSDN density, the solutions converge to deterministic hypervisor location(s). As $|M|$ increases, more vSDN networks are generated with a possibly large geographical distribution. Since these vSDNs are spread with a higher probability among the physical network, the topology of the physical network determines an upper bound for the latencies. For instance, when minimizing the maximum latency, the hypervisor locations are mostly determined by the long links from east to west.

To conclude, the vSDN network density has to be considered for the hypervisor placement. In case of higher densities, less positions may have to be considered as potential hypervisor locations under the given input procedure. This observation can be used to predetermine a set of potential hypervisor locations, i.e., to decrease the size of the set H. As



(a) Maximum latency (b) Average latency



(c) Average Maximum latency (d) Maximum Average latency

Fig. 4. Each metric is evaluated while varying the model objective [L_{max}, L_{avg}, $L_{avg,max}$, $L_{max,avg}$] ($Q = 1, |M| = [1, 140]$).

a result, the runtime of online algorithms for the HPP might be improved, which is considered as future work.

C. What are the trade-offs between the objective metrics?

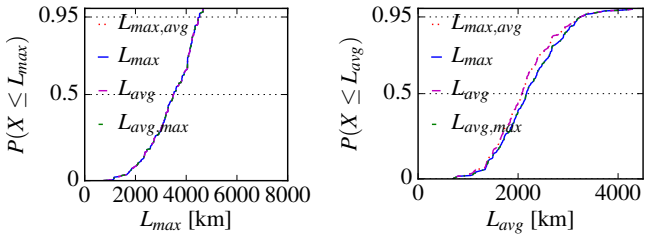
To show the trade-offs between the four objective metrics in more detail, we calculate the values of the other three metrics for each optimization of an objective accordingly. Figures 4(a)-(d) show the average of all metrics with 95% confidence intervals. In general, optimizing the model for a particular metric also leads to the best solution with respect to the metric. For instance, optimizing maximum latency achieves the best solution for L_{max}, while optimizing average latency achieves the best solution for L_{avg}. This proofs the correctness of the proposed model.

In Figure 3 we have observed that the hypervisor locations for the different objectives, e.g., for average-maximum latency and average latency, are overlapping. As a result of the geographically overlapping locations, Figure 4(a)-(d) show the same behavior with respect to all metrics. The results of optimizing average-maximum and average latency are close together for all latency metrics. Respectively, the results for optimizing maximum and maximum-average are close together for all metrics as well.

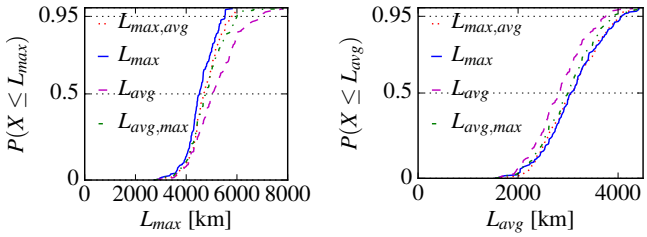
To investigate this behavior in more detail, Figures 5(a)-(f) show the cumulative distribution function for L_{max} and L_{avg} for all objectives. We compare the behavior for $|M| = 1$, $|M| = 3$, and $|M| = 140$ between each objective.

For $|M| = 1$, all models have the same latency results for the maximum latency as shown in Figure 5(a). This means that for all models the hypervisor location is placed on the path having the maximum latency for the particular vSDN network. For the average latency, optimizing for average and maximum-average latency leads to better results. Here, the model optimizes for all paths and does not stop when the maximum latency of one path is reached.

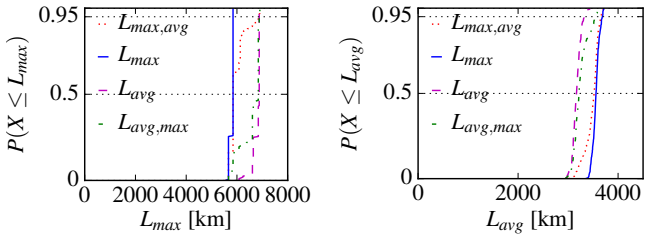
Figures 5(c)-(d) already show a trade-off between the models for $|M| = 3$. In particular for the objectives maximum



(a) Maximum latency, $|M| = 1$ (b) Average latency, $|M| = 1$



(c) Maximum latency, $|M| = 3$ (d) Average latency, $|M| = 3$



(e) Maximum latency, $|M| = 140$ (f) Average latency, $|M| = 140$

Fig. 5. Metrics comparison with all objectives for one hypervisor instance.

latency and average latency, a clear gap exists. In contrast to $|M| = 1$, optimizing for multiple vSDNs leads to different effects. In detail, optimizing for maximum and maximum-average as well as optimizing for average and average-maximum show results that are close together with respect to L_{max} and L_{avg}.

For $|M| = 140$, as shown in Figure 5(e)-(f), a clear trade-off between optimizing for maximum or maximum-average and average or average-maximum can be observed. Furthermore, the resulting latencies are varying less for $|M| = 140$ when comparing to $|M| = 3$. In particular for L_{max}, the optimization leads to a small number of latency values, as indicated by the vertical process of the CDF. We can conclude that the trade-offs between the optimizations of the different metrics depend on the density of the vSDN networks, i.e., their number and their locations. Furthermore, with increasing density, the values of the metrics are varying less among all objective optimizations.

D. What is the solving runtime for different objective metrics?

Figure 6 shows the solving runtime when optimizing for each objective. For an increasing vSDN density, the runtime for all objectives increases and a gap between the different models can be observed. Optimizing for an average-maximum latency shows the largest runtime, while optimizing for the average latency shows the lowest runtime. In general, one factor for the different runtime is the number of constraints needed for each objective. Minimizing the average-maximum latency and maximum latency results in the highest number of constraints. The

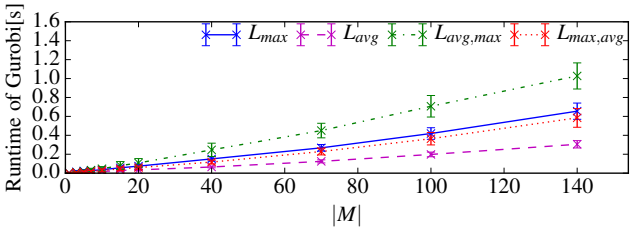


Fig. 6. Solution runtime for different objectives ($Q = 1, |M| = [1, 140]$)

other factor is the complexity of the objective. For example, the objective for optimizing maximum latency consists of only a single value. In contrast, the objective for optimizing average-maximum latency, which has the worst runtime, considers each vSDN additionally. While optimizing for maximum latency and maximum-average latency show comparable runtime and latency performance, both are candidates for optimizing for maximum latency solely.

VI. Conclusion

In this paper, we initiate the study of the placement of SDN network hypervisor instances that enable virtual SDN networks (vSDNs). We call this the network hypervisor placement problem (HPP). A good hypervisor placement is essential for the design of virtual SDN networks as it impacts the control plane latency of the vSDNs. Poor control plane performance for SDN networks can lead to state inconsistency and scalability issues. We provide an initial optimization model to determine the number of needed hypervisor instances and their respective network locations. Additionally, we introduce four metrics covering the latency property of SDN and network virtualization. Our results show performance trade-offs when optimizing for different latency metrics. Furthermore, the number and locations of vSDNs, what we call vSDN density, also impacts the optimization results in terms of hypervisor placement and resulting latency values. Our basic study outlines the importance of fundamental aspects when optimizing virtualized SDN environments. For future work, we are currently analyzing the impact of placing multiple hypervisor instances.

Acknowledgment

This work is part of a project that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets).

References

[1] T. Anderson, L. Peterson *et al.*, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[2] N. McKeown, T. Anderson *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, Mar. 2008.

[3] 5G Initiative Team, "NGMN 5G Initiative White Paper," Feb. 2015, https://www.ngmn.org/uploads/media/NGMN-5G-White-Paper-V1-0.pdf.

[4] R. Sherwood, J. Naous *et al.*, "Carving research slices out of your production networks with OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, p. 129, Jan. 2010.

[5] A. Blenk, A. Basta *et al.*, "HyperFlex: An SDN Virtualization Architecture with Flexible Hypervisor Function Allocation," in *Proc. IFIP/IEEE IM*, Ottawa, Canada, 2015, pp. 397–405.

[6] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. INM/WREN'10*, Berkeley, CA, USA, 2010, pp. 3–3.

[7] A. Tootoonchian, S. Gorbunov *et al.*, "On controller performance in software-defined networks," in *Proc. USENIX Hot-ICE'12*, Berkeley, CA, USA, 2012, pp. 10–10.

[8] D. Levin, A. Wundsam *et al.*, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proc. ACM HotSDN'12*, Helsinki, Finland, 2012, pp. 1–6.

[9] B. Heller, R. Sherwood *et al.*, "The controller placement problem," in *Proc. ACM HotSDN'12*, Helsinki, Finland, 2012, pp. 7–12.

[10] Y. N. Hu, W. D. Wang *et al.*, "On the placement of controllers in software-defined networks," *Journal of China Universities of Posts and Telecommunications*, vol. 19, no. SUPPL. 2, pp. 92–97, Oct. 2012.

[11] Y. Jimenez, C. Cervelló-Pastor *et al.*, "On the controller placement for designing a distributed SDN control layer," in *Proc. IFIP Networking*, Trondheim, 2014, pp. 1–9.

[12] H. K. Rath, V. Revoori *et al.*, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," in *Proc. IEEE WoWMoM*, Sydney, NSW, 2014, pp. 1–6.

[13] G. Yao, J. Bi *et al.*, "On the Capacitated Controller Placement Problem in Software Defined Networks," *IEEE Communications Letters*, vol. 18, no. August, pp. 1339–1342, Aug. 2014.

[14] A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks," *IEEE Communications Letters*, vol. 19, no. 1, pp. 30–33, 2015.

[15] P. Xiao, W. Qu *et al.*, "The SDN controller placement problem for WAN," in *Proc. IEEE/CIC ICCC*, Shanghai, 2014, pp. 220–224.

[16] M. Guo and P. Bhattacharya, "Controller Placement for Improving Resilience of Software-Defined Networks," in *Proc. ICNDC2013*, Los Angeles, CA, USA, 2013, pp. 23–27.

[17] Y. Hu, W. Wendong *et al.*, "Reliability-aware Controller Placement for Software-Defined Networks," in *Proc. IFIP/IEEE IM*, Ghent, 2013, pp. 672–675.

[18] Y. Hu, W. Wang *et al.*, "On reliability-optimized controller placement for Software-Defined Networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[19] D. Hock, M. Hartmann *et al.*, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in *Proc. ITC*, Shanghai, China, Sep. 2013.

[20] S. Lange, S. Gebert *et al.*, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[21] A. Klose and A. Drexl, "Facility location models for distribution system design," *European Journal of Operational Research*, vol. 162, no. 1, pp. 4–29, 2005.

[22] H. Pirkul and V. Jayaraman, "A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution," *Computers & Operations Research*, vol. 25, no. 10, pp. 869–878, 1998.

[23] J. F. Campbell, "Integer programming formulations of discrete hub location problems," *European Journal of Operational Research*, vol. 72, no. 2, pp. 387–405, 1994.

[24] A. Fischer, J. F. Botero *et al.*, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, Jan. 2013.

[25] M. Demirci and M. Ammar, "Design and analysis of techniques for mapping virtual networks to software-defined network substrates," *Computer Communications*, vol. 45, pp. 1–10, Jun. 2014.

[26] M. C. Luizelli, L. R. Bays *et al.*, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE IM*, Ottawa, Canada, 2015.

[27] S. Knight, H. X. Nguyen *et al.*, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, pp. 1765–1775, 2011.

[28] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com