SDN Hypervisors: How Much Does Topology Abstraction Matter?

Nemanja Đerić, Amir Varasteh, Arsany Basta, Andreas Blenk, and Wolfgang Kellerer
Chair of Communication Networks, Department of Electrical and Computer Engineering
Technical University of Munich, Germany
Email: {nemanja.deric, amir.varasteh, arsany.basta, andreas.blenk, wolfgang.kellerer}@tum.de

Abstract—SDN network hypervisors realize the virtualization of software-defined networks. They intercept the control path between tenant controllers and their respective virtual Software-Defined Networks (SDN). Over-utilizing SDN hypervisor resources (i.e., CPU) can degrade the control plane performance of the tenants. Although many hypervisor proposals exist, a detailed performance modeling of SDN hypervisors is missing in literature. A precise modeling of the required SDN hypervisor resources, however, is crucial for predictable and reliable operation of virtual software-defined networks. In this paper, we measure and evaluate how topology abstraction can affect the SDN hypervisor CPU utilization. We consider two topology abstraction cases: the (1) transparent and (2) big-switch abstraction. Our measurements taken from a real testbed indicate that the big-switch abstraction can reduce the SDN hypervisor CPU utilization up to $\sim 4 \times$. Further, we evaluate different functions to model the SDN hypervisor CPU utilization based on our measurement results. Our evaluations show that a polynomial function provides the lowest fitting error. Motivated by our measurements, we conduct a first-step investigation of the impacts of topology abstraction on the Virtual Network Embedding (VNE) problem. Our initial simulation-based evaluations indicate that different topology abstraction procedures impact the results of the VNE problem.

Index Terms—Network virtualization, Software-Defined Networking, Topology abstraction, Virtual network embedding

I. INTRODUCTION

Serving multiple different applications requiring high quality of service in an isolated manner is a prerequisite for future communication network architectures [1], [2]. Network Virtualization (NV) promises isolated sharing of a physical infrastructure among multiple tenants by slicing the data and control planes. In Software-Defined Networking (SDN), the virtualization is typically realized using an additional middle layer; an SDN hypervisor is situated between tenant SDN controllers and the physical data plane [3]. SDN hypervisors translate control plane messages of tenants and ensure full isolation between the tenants. They also hide the underlying unused infrastructure as part of slice (topology) abstraction [4].

In this paper, we focus on topology abstraction which enables tenants to request an arbitrary virtual topology and not only a subset of the physical network. For instance, a tenant can request a big-switch abstraction, where the whole physical topology is represented as one big virtual switch. In the case of big-switch abstraction, an SDN hypervisor has to take over the management of the whole physical representation of the Virtual Network (VN), e.g., setting up paths for forwarding data plane messages between the virtual ports of a big switch. Big switch abstraction simplifies the network management tasks for tenants [4], [5]. Besides, hypervisors can gain additional optimization possibilities, as abstraction allows optimizing the network resources independent of the tenants. For example, hypervisors can freely re-route demands between the ports of a big-switch as long as tenant constraints are fulfilled [6].

However, incorrect provisioning of hypervisor CPU can severely affect the control/data plane performance of tenants [4], [7]. Overloading SDN hypervisors can increase the control plane message processing time, which in turn can increase the flow set-up time of tenants. As the literature does not investigate deeply the effect of abstraction, we evaluate whether different abstraction levels might lead to different CPU performance profiles in this paper. To achieve this, we measure the effects of two different topology abstraction policies on the control plane resources, i.e., SDN hypervisor CPU. Furthermore, we model and evaluate the observed measurements using multiple different fitting functions.

In addition, as future outlook, we show the initial insights of including different abstraction policies in the Virtual Network Embedding (VNE) problem. We believe that adding control plane constraints to the VNE problem renders it to be more realistic. Moreover, it contributes towards realizing full NV — integrating SDN to actually provide tenants with full programmability of their virtual network resources.

II. RELATED WORK

Slice Abstraction: FlowVisor is the first proposed SDN hypervisor [3] that provides abstraction of physical switch ports, i.e., only the physical ports containing the tenants hosts are shown. Since FlowVisor acts like a transparent proxy, it is unable to abstract the intermediate switches. In contrast to FlowVisor, OpenVirtex [8] provides arbitrary topology abstractions, with a limitation that one physical switch cannot be represented as two virtual switches to the same tenant. Further, in [9], authors presented a virtualization layer (VL) developed on the ONOS controller platform [10], which also supports arbitrary topology abstractions. The platform was evaluated in terms of processing time in [11]. Unlike most of the hypervisors, CoVisor [5] enables multiple controllers to...
cooperate on the management of the same data plane traffic by making use of topology abstractions.

**SDN Hypervisor CPU Estimation.** Data plane performance in software-defined networks can be influenced by the control plane performance of SDN controllers [7]: e.g., increased delay in the control plane channel can increase flow setup time. Accordingly, offline benchmarks of SDN hypervisor are performed in order to correlate the number of OF messages and SDN hypervisor CPU utilization [3], [8]. To avoid long offline benchmarks, online machine learning algorithms were proposed in [12]. The algorithms were extended in [13] to support environments with varying CPU resources. However, how topology abstraction affects the SDN hypervisor resources has not yet been discussed in literature.

**Virtual Network Embedding.** The realization of virtualization in SDN provides demands to consider a new dimension: the control plane resources such as SDN hypervisor CPU. This renders new challenges to the Virtual Network Embedding (VNE) problem [14], [15]. To the best of our knowledge, the impact of different SDN hypervisor functions on the VNE problem has not yet been explored in literature.

### III. Topology Abstraction Measurements

In this section, we measure and model the impact of two topology abstraction cases on the SDN hypervisor CPU utilization. The goal of our measurement is to answer the following question: *Does topology abstraction produce an impact on the SDN hypervisor CPU utilization?*

#### A. Setup

Fig. 1a illustrates the measurement setup. We use three PCs to run an SDN controller, an SDN hypervisor, and the data plane network. PC1 emulates the SDN controller by using the SDN benchmarking tool perfbench [16], in order to generate OF FlowModAdd messages with a variable rate. PC2 hosts a Virtual Machine (VM) that runs the SDN hypervisor OpenVirteX [8]. OpenVirteX enables tenants to request arbitrary topologies. However, a limitation is that one physical switch cannot be represented as two virtual switches to one tenant. A CPU monitored on the VM reports on the CPU utilization of OpenVirteX. Finally, PC3 emulates the data plane network with Mininet [17].

**B. Scenario**

In order to investigate if there is any impact of topology abstraction, we construct a simple data plane topology with two hosts H-1 and H-2 connected as a line topology, with \( k \) switches between them (See Fig. 1b). The VN is established between the two hosts and spans all the corresponding physical switches and links as in Fig. 1b. Fig. 1c shows an example of a transparent operation (i.e., no abstraction), while Fig. 1d gives an example of a big-switch abstraction.

**Process:** In OpenFlow (OF) [18], the OF FlowModAdd message is used to add forwarding rules to switches. Thus, in order to establish one traffic flow between the two data plane hosts \( H_1 \) and \( H_2 \), each switch on the path has to receive at least one OF FlowModAdd message. Hence, in total, \( k \) OF FlowModAdd messages are sent by the SDN hypervisor on the southbound interface (SBI) for both abstraction cases. However, the situation on the northbound interface (NBI) differs based on the topology abstraction. In case of transparent abstraction (Fig. 1c), the SDN controller has to generate \( k \) OF FlowModAdd messages towards each switch, while the SDN hypervisor has to forward the messages to the corresponding physical switches. Thus, OF FlowModAdd message rates on the SDN hypervisor NBI \( \lambda_A \) and SBI \( \lambda_B \) are the same (i.e., in Fig. 1a, \( \lambda_A = \lambda_B \)). On the other hand, in the big-switch abstraction case (Fig. 1d), the whole data plane network is abstracted, thus, the SDN controller has to generate only one OF FlowModAdd message to establish the same traffic flow. In this case, the SDN hypervisor has to find a physical route between the virtual ports, and to translate one northbound OF FlowModAdd into \( k \) OF FlowModAdd southbound messages towards each switch on the physical path (thus, in Fig. 1a, \( \lambda_B/\lambda_A = k \)).

**Parameter Settings:** We compare the CPU utilization of OpenVirteX for both topology abstraction cases. We vary the number of switches between the hosts, \( k \in \{2...10\} \), and the data plane flow request rates between the two hosts, \( f = \{10..100\} \). The length of one measurement instance is 90 seconds, where perfbench generates OF FlowModAdd messages, corresponding to the data plane flow rate request. The CPU monitor gathers CPU utilization samples of the VM hosting the OpenVirteX instance every 0.5 seconds. The samples are represented as in percentages of the cores used; e.g., 200% corresponds to two cores being utilized. Notably, we discard the first 5 seconds and the last 5 seconds of each measurement run due to avoid effects from transient phases.

**C. Results**

CPU utilization measurements for \( k = 5 \) and \( k = 10 \) number of switches are shown in Fig. 2a and 2b, respectively. Two box plot samples are shown for each flow rate value (values on x-axis). In each figure, the left box-plots represent the transparent abstraction corner case, while the right ones represent the big-switch case.

For both abstraction cases, increasing the data plane flow rate increases the SDN hypervisor CPU utilization. This is
due to the fact that the number of messages on NBI and SBI are increased, thus, OpenVirteX has to process a higher number of messages. Furthermore, it can be observed that the increase of CPU utilization for the transparent case is much more pronounced. Since the message rate on the SBI is the same for both abstraction cases, it can be concluded that forwarding $k \times f$ messages in the transparent abstraction case requires more CPU resources than calculating physical routes and translating $f$ messages in the big-switch abstraction case.

Moreover, according to the big-switch abstraction case for $k = 5$ (the blue line in Fig. 2a) and $k = 10$ (the blue line in Fig. 2b), it can be seen that the CPU utilization difference is not drastic. Since the messages rates on the NBI in both abstraction cases are the same, it can be concluded that the number of messages on the SBI produces a smaller impact on the CPU utilization, comparing to the number of messages on the NBI.

D. Modeling

Based on our measurements, we suspect that the CPU utilization depends either linearly or polynomially on the required data plane flow rate $f$, the number of switches on the path $k$, and the requested abstraction level $l$. Thus, we formulate linear, quadratic and 3rd order polynomial functions to fit the CPU utilization, as follows:

$$g_{lin}(f, k, l) = c_0 + c_1 f k$$

$$g_{qua}(f, k, l) = c_0 + c_1 f k + c_2 (f k)^2 + c_3 (f k)^3$$

$$g_{pol}(f, k, l) = c_0 + c_1 f k + c_2 (f k)^2 + c_3 (f k)^3$$

where $c$ represents coefficients in the equations. The parameter $l$ is the requested abstraction level which represents the ratio of virtual switches on the virtual path and physical switches on the corresponding physical path. For the big-switch abstraction case, there are one virtual switch and $k$ physical ones, hence $l = 1/k$. In the transparent case, $l = k/k = 1$. Therefore, the multiplications $f k$ and $f k$ actually represent the NBI and the SBI \texttt{OF\_FlowModAdd} message rates, respectively.

Using the \texttt{scipy} Python library, we take the main workload CPU utilization values and find the best fitting coefficients in Eqs. (1,2,3). Table I contains all of the corresponding coefficient values. It shows the average relative errors for all CPU values and the average relative errors for the samples with CPU utilization higher than 30% (Error30). Fig. 3 shows the measured CPU utilization and all of the corresponding models for the transparent case with $k = 10$ switches between the end hosts. As it can be seen in Fig. 3 and Table I, among

---

**TABLE I: Values of Modeling Parameters**

<table>
<thead>
<tr>
<th>Model</th>
<th>Linear</th>
<th>Quadratic</th>
<th>3rd Order Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_0$</td>
<td>5.8956</td>
<td>7.5945</td>
<td>5.4317</td>
</tr>
<tr>
<td>$c_1$</td>
<td>0.0665</td>
<td>0.0345</td>
<td>0.0418</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0.0215</td>
<td>4.5688×10^{-8}</td>
<td>1.2570×10^{-8}</td>
</tr>
<tr>
<td>$c_3$</td>
<td>-</td>
<td>0.0251</td>
<td>2.1590×10^{-8}</td>
</tr>
<tr>
<td>$c_4$</td>
<td>-</td>
<td>-9.0565×10^{-8}</td>
<td>0.0497</td>
</tr>
<tr>
<td>$c_5$</td>
<td>-</td>
<td>-7.2677×10^{-8}</td>
<td>0.0251</td>
</tr>
<tr>
<td>$c_6$</td>
<td>-</td>
<td>-4.2753×10^{-8}</td>
<td>0.0251</td>
</tr>
<tr>
<td>Error</td>
<td>12.29%</td>
<td>10.87%</td>
<td>10.22%</td>
</tr>
<tr>
<td>Error30</td>
<td>7.55%</td>
<td>5.78%</td>
<td>4.49%</td>
</tr>
</tbody>
</table>

---

Fig. 2: CPU utilization with respect to the flow rate between two hosts when there are $k$ number of switches between them. Left column of each box plot represents the case when VN is transparently embedded, while the right column box plots represents if the VN is embedded using big-switch, fitted with 3rd order polynomial function.

Fig. 3: Mean measured CPU utilization for the transparent case with $k = 10$ switches in the data plane and the corresponding estimation models.

Fig. 4: Estimation of CPU utilization based on the presented model for both abstraction cases, i.e., the transparent (no) abstraction and the big-switch abstraction.
all functions, the 3rd order polynomial fits the best, but the error is not significantly lower compared to the other models. Therefore, Fig. 2 used the 3rd order polynomial model for fitting the CPU. From Table I and Fig. 3, it can generally be seen that the models actually perform worse for the lower CPU utilization. It can be observed that increasing either the number of switches or the required flow rate increases the CPU utilization.

IV. FUTURE OUTLOOK

A. VNE and SDN hypervisor provisioning

As future outlook, and as a motivation for the measurements of different SDN hypervisor functions, we provide some initial insights on VNE optimization and control plane resource provisioning in SDN. For this, we extend the VNE problem formulation and definitions from [19] to include the SDN hypervisor control plane resource constraints. In our simulations, we use three different standard topologies: Abilene [20], Internet2 [21], and Germany50 [22]. We present our observations for the VNE problem with the objective to maximize the acceptance ratio.

1) SDN Hypervisor CPU Provisioning: In this part, we simulate the embedding of 100 Virtual Network Requests (VNR) with the same data plane requirements. The VNs use either the transparent abstraction or the big-switch abstraction in the three topologies for 10 times. Fig. 5a shows the required SDN hypervisor CPU resources for each of the topologies. It can be observed that the big-switch topology abstraction requires much lower amount of CPU resources in all topologies. The biggest difference is observed for Internet2 topology, where the big-switch abstraction case requires 50% less resources than the transparent one. It can also be seen that the Abilene topology requires the least amount of CPU resources, as it is the smallest network. Thus, the paths are typically shorter and require fewer number of control plane messages in order to be established.

2) Impact of Topology Abstraction on Objective Function: We vary the total available SDN hypervisor CPU resources from 1 to 30 cores. Fig. 5b depicts the values of objective function for two cases: with and without including the topology abstraction effects in the Internet2 topology. The observed results for the other two topologies follow the same trend; hence, we omit the corresponding figures.

According to Fig. 5b, if the SDN hypervisor resources are overprovisioned (e.g., there are more than 25 available cores), the control plane constraints do not affect the embedding as there are enough control plane resources to accommodate all of the VNRs. Here, the data plane becomes a bottleneck and constraints the embedding of VNs. However, if the SDN hypervisor resources are low (less than 25 cores), the data plane constraints do not affect the embedding. As a consequence, the embedding is mainly affected by the control plane constraints. Here, an incorrect estimation of required SDN hypervisor resources could produce high relative errors in the provisioning process — as a result, tenants might perceive VNs with unpredictable control.

B. Discussion of Initial Topology Abstraction Measurement

In this paper, we focus only on evaluation of topology abstraction effects on a line topology. As a future direction, we plan to investigate more complex topologies in order to fully understand the topology abstraction behavior.

V. CONCLUSION

In this paper, we showed how the correlation between the data plane requirements and the topology abstraction affects the SDN hypervisor CPU utilization. Firstly, in our testbed, we measured the SDN hypervisor CPU utilization for two topology abstraction cases, (1) transparent (no topology abstraction), and (2) big-switch on a line topology. Our measurements indicated the impact of abstraction, as embedding of VNs with big-switch abstraction actually requires less control plane resources than embedding the ones with the transparent abstraction. This effect comes from the difference in the OF message rates on the SDN hypervisor’s NBI (which scales with the number of switches on the path). For instance, transparent abstraction of the path with 10 switches requires around 4× more CPU resources than the big-switch abstraction. We further showed that the corresponding SDN hypervisor CPU measurements can be modeled with 3rd order polynomial function with the average relative error of around 10%.

We also presented a future outlook containing the initial analysis of topology abstraction impacts on the VNE optimization and control plane provisioning in SDN. Initial results indicate that including the topology abstraction in the VNE problem can improve the acceptance ratio up to ∼ 20%. Therefore, the effects of SDN hypervisor functions should not be neglected when provisioning the control plane resources and solving the VNE problem.

ACKNOWLEDGMENT

This work has been performed as part of the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1) and it has been partially funded by the German Research Foundation (DFG) under the grant number KE 1863/8-1.
REFERENCES


