Model-based Performance Predictions for SDN-based Networks: A Case Study

Stefan Herrnleben[1], Piotr Rygielski[2], Johannes Grohmann[1], Simon Eismann[1], Tobias Hoßfeld[1], and Samuel Kounev[1]

[1] University of Würzburg, Würzburg, Germany
{firstname.lastname}@uni-wuerzburg.de
[2] D4L data4life gGmbH, Potsdam, Potsdam, Germany
piotr.rygielski@data4life.care

Abstract. Emerging paradigms for network virtualization like Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) form new challenges for accurate performance modeling and analysis tools. Therefore, performance modeling and prediction approaches that support SDN or NFV technologies help system operators to analyze the performance of a data center and its corresponding network. The Descartes Network Infrastructures (DNI) offers a high-level descriptive language to model SDN-based networks, which can be transformed into various predictive modeling formalisms. However, these modeling concepts have not yet been evaluated in a realistic scenario.
In this paper, we present an extensive case study evaluating the DNI modeling capabilities, the transformations to predictive models, and the performance prediction using the OMNeT++ and SimQPN simulation frameworks. We present five realistic scenarios of a content distribution network (CDN), compare the performance predictions with real-world measurements, and discuss modeling gaps and calibration issues causing mispredictions in some scenarios.

Keywords: Network Modeling · Performance Prediction · Software-defined Networking

1 Introduction

In recent years, data centers became increasingly dynamic due to the wide-spread adoption of virtualization technologies [10]. Virtual machines, data, and services can be offered on-demand and shared as well as migrated between different physical hosts to optimize resource utilization and hence costs while enforcing service-level agreements (SLAs). However, this forms new challenges for accurate and timely performance analyses of these virtualized units and the resulting data centers [3]. In addition to the virtualization of compute resources, the network infrastructures shift towards virtualization as well, with the emergence of paradigms like Software-Defined Networking (SDN) and Network Functions Virtualization (NFV). Therefore, performance modeling and prediction approaches

that support SDN or NFV technologies help system operators to analyze the performance of a data center and its corresponding network.

There exist multiple different modeling formalism to represent data center networks, like, domain-specific simulation models, stochastic Petri nets, queueing networks, and stochastic process algebras. However, modeling with a given formalism requires to understand the meta-model and the usual modeling steps of the respective approach. Thus, specific knowledge and experience with multiple modeling formalisms are required to benefit from the variety of their characteristics. Usually, such knowledge and experience is missing or limited to a single modeling formalism.

In [20], we introduce a modeling approach that models the network using DNI (Descartes Network Infrastructures), a high-level descriptive modeling language. A DNI model can be transformed to multiple predictive models, which enables the application of various modeling and analysis approaches without requiring in-depth expertise in the respective modeling formalisms. Furthermore, we extended the DNI to also capture SDN-based network solutions [22]. While the modeling concept of non-SDN networks was already evaluated in [21], the SDN models have not been evaluated yet.

In this paper, we present an extensive evaluation of the DNI, its proposed model-to-model transformations, and two simulation frameworks OMNeT++ [28] and SimQPN [16] in the context of a realistic case study. This case study models file download scenarios within a content distribution network (CDN). We compare their predictions with network measurements and use the results to identify modeling gaps and draw conclusions about directions for future work. We present five different scenarios, covering non-SDN-based networking, SDN-based networking using hardware tables, SDN-based networking using software tables, node virtualization, and routing via the SDN controller. By analyzing the above scenarios, we can compare the simulation techniques as well as the corresponding model-to-model transformations and therefore contribute to a better understanding of the respective techniques.

2 Performance Modeling and Prediction Methodology

In this section, we briefly introduce the main concepts of the *Descartes Network Infrastructure Modeling (DNI) language* and the corresponding performance prediction pipeline. The DNI meta-model is a descriptive model to describe a network infrastructure and covers the three main aspects of every data center network infrastructure: the topology, the configuration, and the traffic. The topology includes all nodes, links, and interfaces, which can either be physical or virtual. Nodes can be end nodes (e.g., virtual machine, server) or intermediate nodes (e.g., switch, router). SDN switches act as both intermediate and end nodes, as they are forwarding devices but also interact with the SDN controller. Figure 1 depicts an example of the infrastructure modeling. For every element in the model, the performance-relevant parameters are described. The performance for each element has to be specified according to the vendor datasheet or by con-

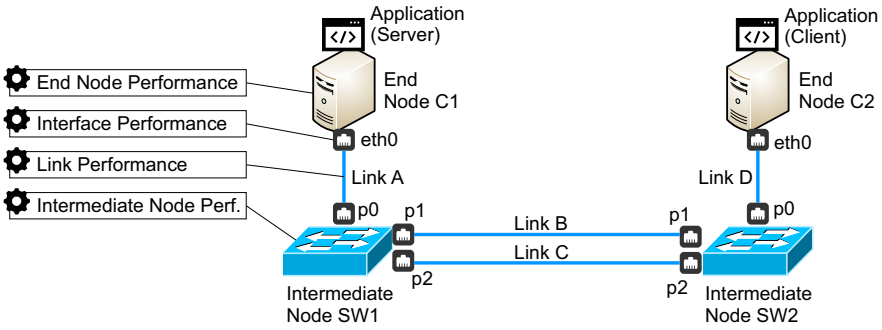


Fig. 1: Example DNI infrastructure model.

ducting tailored measurements. The DNI network modeling language considers different forwarding performance characteristics for SDN switches, depending on whether the packet is processed using hardware or software flow tables [23]. Sending the packet headers of unknown flows from an SDN switch to the SDN controller is also supported by DNI with a separate performance description. Traffic is modeled in DNI as flows that are linked to communicating applications, which are deployed on end nodes. Each flow has exactly one traffic source – the traffic generator – and multiple possible destinations. Flows can be composed into a workload model that defines the payload size as well as the temporal behavior by supporting sequences, loops, and branches. Lastly, the configuration of a network contains information about routes, protocols, and protocol stacks. The flexible modeling concept of DNI allows defining customized protocols and protocol stacks, including overheads by the data unit headers. The information of the configuration is used to calculate the paths in the topology graph and coarsely estimate the overheads introduced by the protocols. A route in DNI can either be described between a pair of nodes (source and destination) or flow-based, by defining a route for every flow individually. The flow-based routing representation enables the modeling of software-defined networks, which might use the OpenFlow protocol, for example.

The accuracy of the simulation – as well as the simulation time – is primarily dependent on the level of detail of the performance model. As highly detailed models require increased simulation time, they might not be applicable for large data center networks prediction; additionally, they require specifying many low-level parameters, which might be cumbersome for large networks. Therefore, DNI enables modeling in different levels of detail and therefore supports different use cases. Additionally, in order to predict the performance of a model, a transformation into a predictive model has to be performed, as shown in Figure 2. The transformation from the descriptive model into a predictive model is performed by adapters, which can be supplemented by transformations into additional prediction models. Currently, DNI supports transformations to Queueing Petri Nets (QPNs) [22] and to OMNeT++generic [20]. The generated models can be simulated using OMNeT++ [28] and SimQPN [16] (for QPN), respectively. This

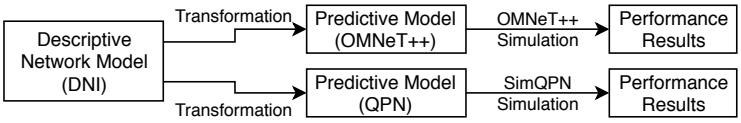


Fig. 2: Workflow from descriptive network model through transformation into a predictive model and simulation to performance results.

enables users to decide on the right simulation framework, depending on the specific requirements (in terms of, e.g., accuracy and simulation time) of the given use-case.

3 Case Study Design

In this section, we describe the base scenario of the case study presented in this work. The presented evaluation uses a file download scenario, where different file resources are requested by multiple clients over a specified network. As the file resources are redundantly located on the servers, client requests may be handled by one or more servers and thus provide the resources from an optimal network location. Typical examples for such distributed resource requests within a data center are cluster file systems or software-defined storage [5, 29, 7]. Content distribution networks (CDNs) [14, 19] for files or streams are another example of such distributed file requests which are frequently used both inside and outside a data center. A client can request single or multiple resources, either as batch download or one by one, with a deterministic or exponentially distributed pause between each of the individual requests. The presented experiments are executed leveraging the L7sdntest software [25], which works similar to Uperf [26]. L7sdntest includes specialized SDN features, like support of the SNMP and OpenFlow protocols, the signaling between clients and servers to the SDN controller, and the management of SDN flow tables by the SDN controller. Additionally, the L7sdntest provides a central experiment controller to control complex experiments.

The structure of the used network is based on a representative data center layout and is visualized in Figure 3. A redundant pair of distributions switches `SW10` and `SW00` represent the backbone of the network. The top-of-rack switches `SW40`, `SW41`, `SW42`, `SW43`, and `SW35` are connected to each of these distribution switches. The nodes `C10`, `C11`, `C12`, `C13`, `C17`, `C36`, `C37`, `C38`, and `C39` are connected to the top-of-rack switches and represent data center servers or clients. The role of the nodes can be freely configured, as L7sdntest software implements each client and server in a unified way so that each client has the functionality of a server and vice versa. The SDN controller, deployed on node `C16`, is directly connected to distribution switch `SW00`.

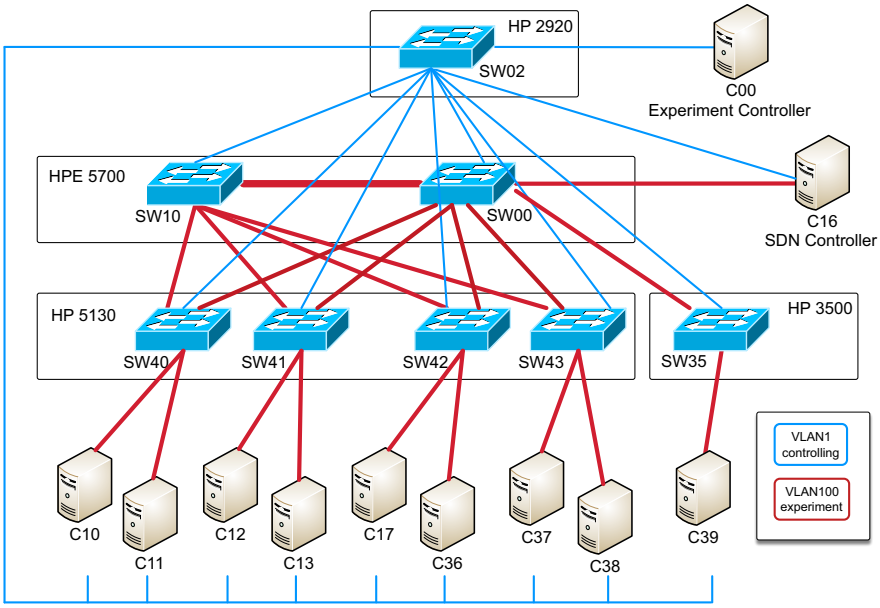


Fig. 3: Experimental testbed used for SDN experiments.

3.1 Hardware Testbed

For the hardware testbed, nine commodity servers (`C10`, `C11`, `C12`, `C13`, `C17`, `C36`, `C37`, `C38`, and `C39`) are used. Each server is equipped with a four-core CPU, 32GB of memory, and a 1 Gbps Ethernet network interface. Eight of them are connected to HP 5130 top-of-rack switches `SW40`, `SW41`, `SW42`, and `SW43`, one of them to the `SW35` HP 3500 switch, all with a bandwidth of 1 Gpbs. The topology of the hardware testbed is shown in Figure 3. The HP 5130 run the *Comware* switch operating system, the HP 3500 is controlled by the *ProVision* switch operating system. The top-of-rack switches are connected to the HP 5700 distribution switches `SW10` and `SW00` via 10 Gpbs SFP+ DAC copper cables. The distribution switches are connected with each other using copper QSFP+ DAC with a maximum bandwidth of 40 Gpbs. Depending on the required OpenFlow version, *Ryu* [27] was used for OpenFlow 1.0, and *HP SDN VAN Controller* [9] was used for OpenFlow 1.3. The SDN controller was deployed on the server `C16`. In order to isolate the experiment control traffic from the experiment traffic, two different VLANs and an additional switch `SW02` are used. An experiment controller, deployed on server `C00`, manages the L7sdntest software, requests switch statistics via SNMP over the isolated control traffic network, and stores the results.

3.2 Modeling

Predict the performance of the network, requires building a corresponding DNI network model. As described in Section 2, all servers, switches, network interfaces, and links have to be mapped to their corresponding element in the model. While the servers `C10`, `C11`, `C12`, `C13`, `C17`, `C36`, `C37`, `C38`, and `C39` act as end nodes that generate and consume traffic, the switches `SW10`, `SW00`, `SW40`, `SW41`, `SW42`, `SW43`, and `SW35` are modeled as intermediate nodes which forward traffic. When the network operates in SDN mode, the switches also get the role of end nodes assigned as they communicate with the SDN controller via the Southbound API with a control channel protocol like, e.g., OpenFlow [12]. If virtual machines are used, they are modeled as nodes, each with an own performance description, and are assigned to a physical node. The SDN controller itself is modeled as an application on server `C16`. Network interfaces are also modeled as child entities of clients and servers. The connections between servers and switches are modeled as links with additional performance parameters. All end nodes get a performance description, specifying a software layer delay. The intermediate performance description for the switches defines the forwarding latency, switching capacity, and forwarding bandwidth. For SDN switches, the performance for processing the packets using hardware tables is specified together with the performance of using software tables. Switches of the same type each receive identical performance descriptions. For all network interfaces, the packet processing time and the interface throughput is defined within their performance description. Furthermore, for each link, a propagation delay as well as a maximal supported bandwidth is specified. The link from the server to the top of rack switches are configured by a maximal supported bandwidth of 1 Gpbs and their uplink to the distribution switches with a maximal bandwidth of 10 Gbps. The link between the distribution switches `SW10` and `SW00` is specified using a maximal supported bandwidth of 40 Gbps in the network model. The SDN controller on server `C16` is connected via a 1 Gbps link to the distribution switch `SW00`. In the protocol configuration, the L4 protocols TCP and UDP, with their underlying protocol stack, i.e., IP and MAC, each with their maximum data payload as well as the packet overhead, are modeled.

To transport the packages through the network, corresponding routes are configured. DNI supports two routing mechanisms: In the classical mode, routes are specified on each interface whereby at each interface, one route is marked as the default route. The classical routing mode is used for all non-SDN measurements. For the SDN scenarios, the second routing mechanism of DNI, the SDN flow rules, are used. An SDN flow rule refers to a node, a flow, as well as an SDN controller. A probability configuration attribute specifies whether the packet should be processed using hardware or using software tables. The last part of the model is the definition of the workload. Each request is mapped to a flow and the traffic sources and destinations are specified for all flows. Additionally, the size of the file to be transferred is specified for each flow.

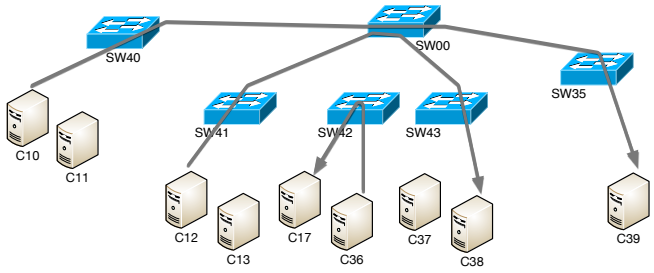


Fig. 4: Flows of server replies (denoted by arrows).

4 Case Study

In this section, we evaluate the prediction accuracy of predictive models obtained in the flexible performance prediction approach for SDN-based data center networks. The approach leverages DNI models instantiated from the DNI metamodel and obtains predictive models using model transformations. For each DNI model, we generate two predictive models using the *DNI-to-OMNeT++generic* and *DNI-to-QPN* model transformations. The generated models are solved with the respective OMNeT++generic and SimQPN solvers. Based on the base scenario introduced in Section 3, we analyze the following five scenarios:

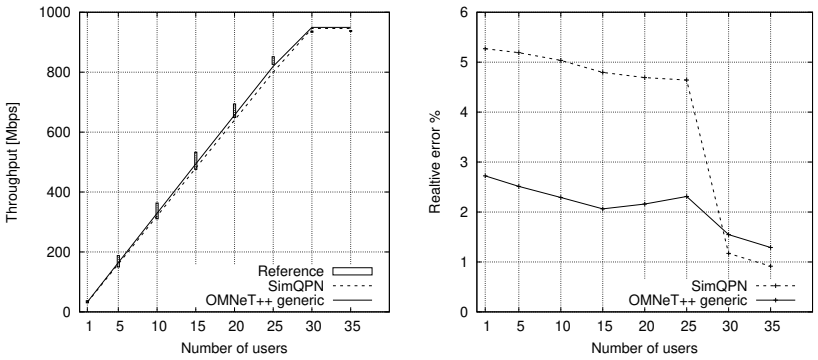
Scenario #1 **Non-SDN Networking**
Scenario #2 **SDN Hardware Tables**
Scenario #3 **Node Virtualization**
Scenario #4 **Software Flow Tables**
Scenario #5 **SDN Controller**

Each scenario consists of at least 30 repetitions of the experiment scenario. A single repetition of an experiment scenario takes between 3 and 60 minutes, depending on the specific scenario. For the measured data, an average throughput is calculated for each second after removing the warm-up and cool-down periods. Finally, the average steady-state throughput, as well as confidence intervals and percentiles, are calculated. Additionally, all models used in this case study are available online[3].

4.1 Non-SDN Networking

In this scenario, three client applications are deployed on servers C39, C38, and C17. The clients request files from predefined servers: C10, C12, and C36. The clients request 100 times the same resource of size 20 MB, where each request is issued every 5 seconds. The breaks between requesting the resources are deterministic. The communication pattern is presented in Figure 4, where

[3] https://gitlab2.informatik.uni-wuerzburg.de/descartes/dni-metamodel/tree/dev/examples/sdn-measurements



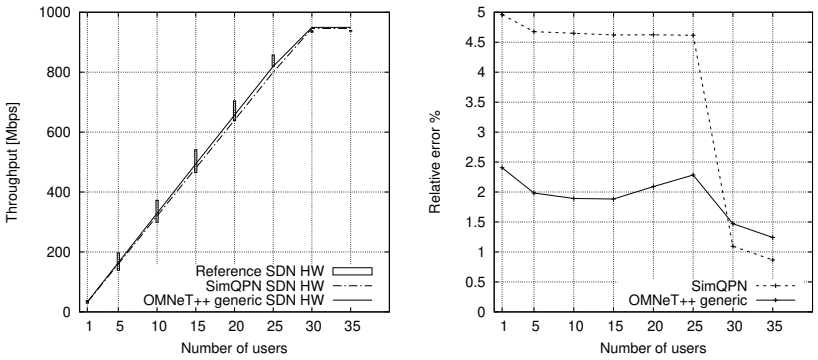
(a) Measured/predicted throughput. (b) Relative prediction error.

Fig. 5: Network throughputof SW35→C39 with non-SDN networking.

each gray arrow represents the reply of a server to the respective client. Each server reacts to the client requests immediately and starts transmission of the requested resource. The three pairs of servers communicate simultaneously and share the network infrastructure with each other. For each communicating pair, we measure the network capacity for 1, 5, 10, 15, 20, 25, 30, and 35 users. These measurements of reference throughput are obtained using SNMP-based monitoring procedures implemented in L7sdntest software.

Modeling For this scenario, the DNI model described in Section 3.2 is used as a base. As this scenario investigates on a non-SDN network, we remove all model elements corresponding to SDN features, such as the SDN controller and the SDN applications on the switches. Additionally, the SDN flow rules are replaced by more coarse-grained interface-based route configurations.

Results In this scenario, we compare the prediction accuracy of two predictive models: OMNeT++generic and QPN. Based on the obtained predictions, the network reaches its maximal capacity at the level of about 25-30 users. The Throughput does not increase beyond the maximum of 942 Mbps for 30 and 35 users. The measured and predicted throughput on a selected network interface (SW35→C39) is presented in Figure 5a, whereas the relative prediction error in Figure 5b. The measurement results are stable and the size of confidence intervals does not exceed 60 Mbps. Both generated predictive models provided very good performance prediction accuracy. OMNeT++generic delivered predictions with lower error (at most 3%) than SimQPN (at most 5.2%). The maximal absolute prediction error was below 40 Mbps for SimQPN and 20 Mbps for OMNeT++generic. For the case of 1-25 users, both solvers underestimated the throughput, whereas, for a fully saturated network, the predictions were overestimating the measured capacity by up to 14 Mbps. Based on these measurements, we conclude that DNI correctly models the structure and performance properties of the traditional network.



(a) Measured/predicted throughput. (b) Relative prediction error.

Fig. 6: Network throughput of SW35→C39 using SDN hardware tables.

4.2 SDN Hardware Tables

For this scenario, the switches from the previous experiment setup are reconfigured to work in SDN mode in order to evaluate the prediction accuracy for SDN-based networks. The SDN switching is based on flow rules that match the IP addresses of the traffic flows. This scenario assumes proactive flow insertion. Therefore, the flow rules are inserted by the SDN controller before the arrival of the traffic. In order to evaluate the performance predictions for hardware SDN modes, we assert that all flow rules are inserted into hardware flow tables of the respective switches and the table capacity is not exceeded.

Modeling The DNI models adapted to SDN hardware mode by adding SDN flow rules and enable the processing of the rules in the hardware tables of the switches. Each SDN node in DNI gets assigned an additional SDN performance description. The official vendor data sheets do not include the performance of SDN, so we use the forwarding performance and other performance-relevant characteristics of the switches that have been published in [24].

Results The measured and predicted throughputs on the network interface SW35→C39 are presented in Figure 6a. These results show that there is no significant difference in performance between the native and SDN hardware mode for the analyzed switches. The network gets saturated for 30 users and offers a maximum throughput of about $942 Mbps$. The maximum relative deviation of offered throughput does not exceed 1%. However, the bounds of confidence intervals differ up to 7% for the measurement with five users. Therefore, the deviations likely stem from measurement errors. The predictions for SDN hardware mode provide almost identical capacity prediction. Both generated predictive models provide high accuracy with maximum prediction errors of 5% for SimQPN and 2.5% for OMNeT++generic. While the error for SimQPN is higher than for OMNeT++generic, we consider the prediction error as acceptable and therefore conclude that DNI provides the capability to model SDN hardware mode.

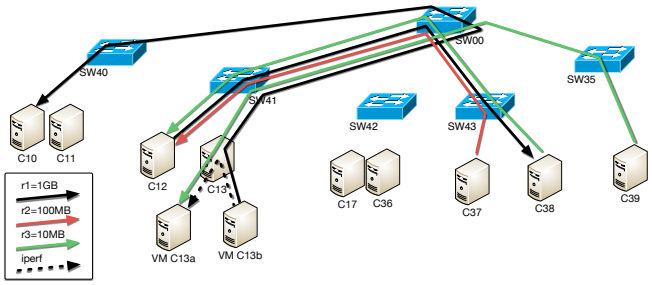


Fig. 7: Experimental testbed and file transfer flows with node virtualization.

4.3 Node Virtualization

SDN networks are often used in conjunction with virtualized compute resources, which results in different network routings. In this scenario, we introduce server virtualization on node C13 and modify the service deployment to obtain a new configuration of flows. Server C13 becomes a Xen [2] hypervisor that hosts two virtual machines C13a and C13b with four CPU cores and 8GB memory each. The L7sdntest services are redeployed to leverage the high bandwidth of the 10Gbps SFP+ links between the switch SW00 and switches SW4x. Additionally, we add multiple users that request the resources of diversified sizes: $r_1 = 1000$, $r_2 = 100$, and $r_3 = 10MB$ respectively (selected arbitrarily). We also configure a new flow of file resources that connects the virtual machines C13b and C13a to investigate the influence of the hypervisor on the network capacity. The full testbed configuration is presented in Figure 7. We use the same measurements procedure as previously for all flows traversing a physical switch. Unfortunately, the SNMP implementation in the Xen hypervisor (node C13) cannot provide stable reports from virtual interface byte counters. Therefore, we measure flow C13b-C13a separately using Iperf [8] to estimate the maximal capacity of the VM-to-VM connection.

Modeling The modeling of the server virtualization consists of defining the VMs, a virtual switch (a bridge running on the hypervisor), and virtual links connecting the VMs to the hypervisor bridge. The bandwidth of links and network interfaces is set to $1000Gbps$ in both predictive models (as an approximation of infinity). The maximal bandwidth of the virtual hypervisor bridge was specified using forwarding delay. The value of forwarding delay was estimated using a controlled experiment and set to $1\mu s$ for the Iperf workload. The experiment was executed under a repeatable constant level of computing load on the SDN controller.

Results The measured and predicted throughputs are presented in Table 1. While both predictive models deliver predictions with low errors, the SimQPN solver performed better than OMNeT++generic and provided up to 2% more accurate predictions. The absolute prediction error of the high-bandwidth link ($SW41 \rightarrow SW00$) was low and did not exceed 40 Mbps on average. Anomalies can be observed for three of the monitored network interfaces. The flows sharing

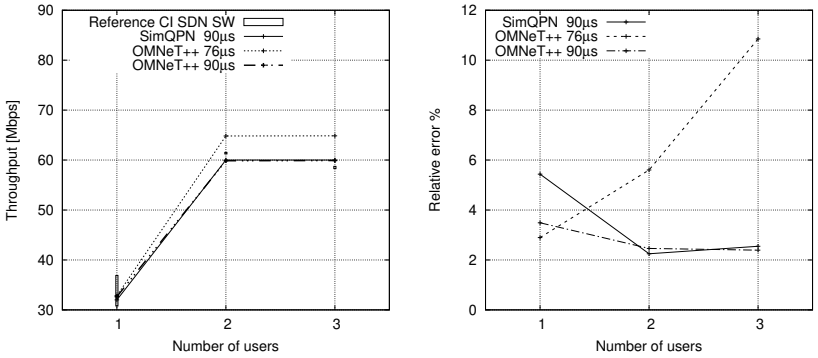
Table 1: Scenario #5: Measured and predicted network capacity.

Measured port	Reference *Mbps*		QPN *Mbps*	Relative error %	OMNeT *Mbps*		Relative error %
	lCI	uCI	avg		lCI	uCI	
Scenario #5A (no flow C13b → C13a)							
SW41 • → SW00	1834	1888	1848	0.7	1879	1924	2.2
SW00 • → SW41	171	196	192	4.9	163	232	7.9
SW41 • → C12	152	174	176	7.8	154	208	11.0
SW41 • → C13	18	22	16	19.6	14	19	17.4
SW00 • → SW40	920	941	924	0.7	940	962	2.2
SW00 → • SW43	917	944	924	0.7	938	961	2.1
SW43 • → C38	917	942	924	0.6	945	954	2.2
Scenario #5B (flow C13b → C13a with *iperf*)							
C13b → • C13	11052	11149	17040	54	16823	17987	57
C13 • → C13a	11052	11149	10878	2.0	9999	10256	8.8

the path $SW00 \rightarrow SW41 \rightarrow C12$ and $C13$ consumed less network resources than predicted by the models. The total consumption of capacity on the link $SW00 \rightarrow SW41$ is expected in theory as maximally $192Mbps$. Despite the ideal prediction of SimQPN, the reference measurements provided larger confidence intervals and thus the average is reported below the expected $192Mbps$. OMNeT++generic, however, predicted a higher variation of the average consumed capacity and thus the prediction is provided with higher accuracy error. This phenomenon propagates further to the links $SW41 \rightarrow C12$ and $SW41 \rightarrow C13$, so higher prediction errors are observed. Note that the absolute prediction errors are low and do not exceed $25Mbps$ and $5Mbps$ for links $SW41 \rightarrow C12$ and $SW41 \rightarrow C13$, respectively. Despite the challenging calibration procedure, the performance predictions for the flow C13b → C13a that are measured using *iperf* return accurate results with 2% and 8.8% prediction error for SimQPN and OMNeT++generic respectively. The flow $C13b \rightarrow C13$, however, is affected by the TCP-UDP modeling gap, that is, the predictive models analyze the traffic in an UDP-fashion, whereas the reference communication runs over TCP and the throughput of the flow is limited by congestion control algorithms according to the bandwidth of a bottleneck resource.

4.4 Software Flow Tables

In this experiment, we reconfigure the SDN switches to forward the incoming traffic based on MAC addresses instead of IP addresses. This forces the $SW35$ switch to install the rules into the software flow table. The rest of the switches install the new rules in the hardware tables. Thus, for the switch $SW35$, we investigate the offered network capacity in the SDN software mode. The other switches contain only hardware flow tables, so their performance in SDN software mode cannot be analyzed.



(a) Measured/predicted throughput. (b) Relative prediction error.

Fig. 8: Network throughput of SW35→C39 using SDN software tables.

Modelling The software SDN forwarding mode is modeled in DNI by disabling the processing of the flow rules in the hardware table and thereby force usage of the software flow tables. The switching capacity for software switching is set to 10.000 packets per second in DNI. This represents the maximal switching capacity offered by the switch operating system. We experimentally estimated the forwarding in SDN software mode to $90\mu s$. Additionally, we also evaluate a forwarding delay of $76\mu s$ (as estimated using the method presented in [22]).

Results Setting the SDN forwarding mode to software switching causes a drastic drop in the offered throughput. The switch $SW35$ is able to deliver maximally 62 Mbps of throughput, which corresponds to 6.5% of the maximal throughput in the SDN hardware mode. There are two main factors that contribute to the observed performance drop. First, the switch operating system limits the maximum switching capacity to 10.000 packets per second. This limit is configurable and can be set to lower values, however for the maximum setting, the switch consumes already almost 90% of its CPU resources (as presented in [24]). Second, the software flow table is usually implemented using general-purpose SDRAM, so the lookup procedure consumes additional time to find a matching rule in the flow table. This incurs additional forwarding delay that needs to be estimated empirically. The measured and predicted throughputs are depicted in Figure 8a, whereas the relative prediction errors in Figure 8b. Both predictive models predict the drastic performance degradation and provide accurate estimates of the maximal network capacity. The OMNeT++generic simulation with forwarding delay $90\mu s$ predicts the average throughput accurately with a prediction error below 4%. SimQPN performed similarly, however the prediction error reaches about 6% for a single user. The alternative method for calibrating the forwarding delay results in higher inaccuracies and the prediction error for OMNeT++ reaches 11%. This can still be considered an acceptable prediction accuracy under the assumption that the forwarding delay was calculated a priori using an analytical formula.

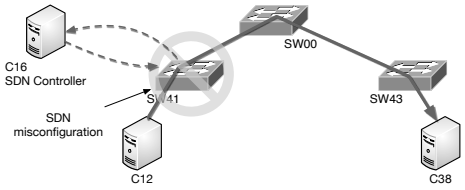


Fig. 9: Experimental testbed and network flows for the SDN controller scenario.

4.5 SDN Controller

In this scenario, we assume that an error in the SDN flow rule configuration causes all switches to misinterpret the rules located in the flow tables and forward all traffic via the SDN controller. This way, we indirectly examine the performance of the SDN controller in handling excessive *packet-in* traffic. The switches are configured to forward to the controller if no rule in the flow tables can be matched. Additionally, the SDN controller application returns the *flow-mod* messages that do not install any rules in the flow tables. Instead, the switch is instructed to forward each packet directly to the proper outgoing port. In this scenario, two servers are communicating via a network path containing three SDN switches, as shown in Figure 9. Next, we enable SDN on switch $SW41$, whereas the other switches work in native mode. The *Ryu* SDN controller is connected directly to SW41 over a dedicated network link. Node C38 represents a single user requesting a 20 MB resource from C12 every five seconds. Unfortunately, L7sdntest could not establish a stable connection between the client and the server. Note that in each of 30 experiment repetitions, the software handles 100 consecutive file transfers and reports an experiment failure if any of the experiment repetition fails. Therefore, we use Iperf to emulate the user behavior for this scenario.

Modelling In this scenario, we build the DNI model similarly as in the previously presented SDN scenario. The modeling consists of defining a flow between nodes C10 and C38 with respective SDN flow rules. But instead of processing flows in hardware or software tables within the data plane, the SDN devices are modeled to forward them to the controller by default. The modeling of an SDN controller faces similar challenges to the calibration of forwarding delay of the SDN switch working in SDN software mode. Therefore, we empirically estimate the per-packet processing delay by the SDN controller application to 8ms.

Results The reference measurements shown in Table 2 originate from three Iperf settings. First, we use the default Iperf command that measures the maximal capacity of the network connection. In this scenario, the average bandwidth of the path connecting node C10 and C38 is low and does not exceed 1.3 Mbps of stable traffic throughput. Both SimQPN and OMNeT++generic overestimate the throughput on link $C12 \rightarrow SW41$ due to the previously discussed differences between TCP and UDP. For the link $SW43 \rightarrow C38$, the high relative errors correspond to an acceptable absolute error of 0.15 Mbps and 1.18 Mbps respectively.

Table 2: Measured and predicted throughput in the SDN controller scenario.

Measured port	Reference *Mbps*	QPN *Mbps*	Relative error %	OMNeT *Mbps*		Relative error %
	avg	avg		lCI	uCI	
Reference: default TCP *iperf*						
C12 → • SW41	1.24	32.00	2481	27.12	38.60	2550
SW43 • → C38	1.24	1.39	12	2.14	2.70	95
Reference: default UDP "*iperf –udp*"						
C12 → • SW41	1.03	32.00	3007	27.12	38.60	3090
SW43 • → C38	1.03	1.39	35	2.14	2.70	135
Reference: modified UDP "*iperf –udp -b 32000000*"						
C12 → • SW41	32.00	32.00	0	27.12	38.60	2.6
SW43 • → C38	2.19	1.39	57.5	2.14	2.70	10.5

Next, we switch the transport protocol to UDP and measure the available bandwidth. This however, does not allow Iperf to send more data than 1 Mbps unless the receiving side confirms successful receptions. The maximal non-interrupted transfer measured by Iperf limited the throughput to 1.03 Mbps. Iperf tries to transmit data with 2 Mbps but it returns to the throughput of 1 Mbps due to high packet losses [8]. The relative and absolute prediction errors are high and therefore opportunities for improvement in future work.
Finally, we force Iperf to send 32 Mbits of data without waiting for the confirmation of the receiving side. The switch SW41 forwards each packet to the SDN controller. After approximately 8ms, the controller replies with a decision regarding the packet forwarding, which is not stored in the flow table. Therefore, each packet is delayed by at least 8ms (plus additional network interface processing delays) and forwarded to the destination in C38. At the receiving end-point, Iperf reported maximal throughput of 2.19Mbps with 130ms jitter and packet loss rate of 92%. Iperf measures the performance at the receiver side until the sender side notifies it that the experiment ends. However, the notification is significantly delayed by the switch and the SDN controller, so it arrives at the receiver later. Due to this additional delay, an additional part of the datagrams queued at the SDN controller arrive at the receiver and are included in the statistics. As the behavior in this scenario is closely related to the modeled behavior, the utilized network capacity has been predicted exactly by SimQPN, whereas OMNeT++generic mispredicted it with 2.6% relative error (0.86 Mbps absolute error). While the relative prediction errors on link $SW43 \rightarrow C38$ were higher —57% for SimQPN and 10.5% for OMNeT++generic — the absolute errors of 0.8 Mbps and 0.23 Mbps can be considered acceptable.

5 Related Work

We group the related work into two clusters. First, we describe all related works modeling the performance of data center networks on an architecture level. Note

that none of the approaches is able to capture SDN-based networks. Second, we list performance modeling approaches capable of modeling SDN-based networks. However, none of those include explicit architecture-level modeling like DNI does.

Architecture-Level Performance Modeling of Data Center Networks Several approaches for modeling networks in data center networks have been proposed. In [30], the authors propose to extend the SDL and UML languages with performance annotations. In [6], the authors propose a modeling approach named Syntony. A similar approach is presented in [18]. Similar to [30], all approaches focus the modeling on the protocol-level. The authors of [13] present a stochastic model for the window dynamics in TCP and investigate the throughput performance of TCP-Tahoe. The I/O path model (IOPm) [17] was designed to model the architecture of parallel file systems. However, none of the above works are capable of modeling SDN networks.

Performance Modeling of SDN-based Networks The authors of [1] propose an analytical performance model based on network calculus. In contrast to our approach, the proposed model does not cover the computing infrastructure, the software architecture, and the hosted applications. The authors of [11] propose a performance model based on queueing theory to evaluate SDN-enabled switches. The evaluation focuses only on OpenFlow-enabled switches and controllers; the scope of the complete data center architecture is missing. Similarly, the authors of [4] focus only on a selected part of SDN-based networks, i.e., the data plane of a switch. The authors of [15] proposed an SDN extension to OMNeT++ simulation based on the INET library. Nevertheless, given the simulation at the protocol-level, their approach focuses on the specific network protocols supported by the INET library and misses the scope of the entire data center.

To the best of our knowledge, the only work considering modeling SDN-based networks at the architecture level is our prior work [22]. However, the prior work fails to exhaustively evaluate the performance of the modeling approach, which is what we are targeting in this work.

6 Conclusion

In this paper, we present an extensive case study evaluating performance predictions of SimQPN and OMNet++ on SDN-based networks. Both simulation models are created by model-to-model transformations based on the Descartes Network Infrastructure (DNI) model. We present five different scenarios, covering non-SDN-based networking, SDN-based networking using hardware tables, SDN-based networking using software tables, node virtualization, and routing via the SDN controller. Our results show that both simulation frameworks deliver comparable and sufficiently accurate predictions for most scenarios. However, we also identify some remaining challenges and open issues. For example, we notice the TCP-UDP-modeling gap. Here, if the predictive models simulate the traffic in an UDP-fashion, while TCP operated by the reference application limits the throughput of the flow using congestion control algorithms. Modeling of congestion control algorithms for network protocols in DNI would close this gap,

which is planned as follow-up work. Additionally, the manual calibration of the forwarding delay parameter for SDN switches is cumbersome and error-prone. In the future, fully automated calibration approaches could be investigated.

Acknowledgements

This work was funded by the German Research Foundation (DFG) under grant No. (KO 3445/18-1).

References

1. Azodolmolky, S., Nejabati, R., Pazouki, M., Wieder, P., Yahyapour, R., Simeonidou, D.: An analytical model for software defined networking: A network calculus-based approach. In: IEEE Global Communications Conference (GLOBECOM). pp. 1397–1402 (Dec 2013)
2. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. SIGOPS Oper. Syst. Rev. **37**(5), 164–177 (2003)
3. Bezemer, C., Eismann, S., Ferme, V., Grohmann, J., Heinrich, R., Jamshidi, P., Shang, W., van Hoorn, A., Villavicencio, M., Walter, J., Willnecker, F.: How is Performance Addressed in DevOps? In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. pp. 45–50 (2019). https://doi.org/10.1145/3297663.3309672
4. Bianco, A., Birke, R., Giraudo, L., Palacin, M.: Openflow switching: Data plane performance. In: Communications (ICC), 2010 IEEE International Conference on. pp. 1–5 (May 2010)
5. Clements, A.T., Ahmad, I., Vilayannur, M., Li, J., et al.: Decentralized deduplication in san cluster file systems. In: USENIX annual technical conference. pp. 101–114 (2009)
6. Dietrich, I., Dressler, F., Schmitt, V., German, R.: SYNTONY: network protocol simulation based on standard-conform UML2 models. In: Proc. of the ValueTools '07. pp. 21:1–21:11 (2007)
7. Donvito, G., Marzulli, G., Diacono, D.: Testing of several distributed file-systems (hdfs, ceph and glusterfs) for supporting the hep experiments analysis. In: Journal of physics: Conference series. vol. 513, p. 042014. IOP Publishing (2014)
8. ESnet: NLANR/DAST : Iperf - the TCP/UDP bandwidth measurement tool. Online, https://iperf.fr/ (Accessed September 2016), https://iperf.fr/
9. HP: Hp SDN controller architecture. Tech. rep., Hewlett-Packard Development Company, L.P. (September 2013)
10. Huber, N., von Quast, M., Hauck, M., Kounev, S.: Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments. In: Proc. of the 1st Int. Conf. on Cloud Computing and Services Science. pp. 563–573 (2011)
11. Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P.: Modeling and performance evaluation of an openflow architecture. In: Teletraffic Congress (ITC), 2011 23rd International. pp. 1–7 (Sept 2011)
12. Jarschel, M., Zinner, T., Hoßfeld, T., Tran-Gia, P., Kellerer, W.: Interfaces, attributes, and use cases: A compass for sdn. IEEE Communications Magazine **52**(6), 210–217 (2014)

13. Kaj, I., Olsén, J.: Throughput modeling and simulation for single connection tcp-tahoe. In: Jorge Moreira de Souza, N.L.d.F., de Souza e Silva, E.A. (eds.) Teletraffic Engineering in the Internet Era, Teletraffic Science and Engineering, vol. 4, pp. 705–718. Elsevier (2001)
14. Kangasharju, J., Roberts, J., Ross, K.W.: Object replication strategies in content distribution networks. Computer Communications **25**(4), 376–383 (2002)
15. Klein, D., Jarschel, M.: An openflow extension for the omnet++ inet framework. In: Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques. pp. 322–329. SimuTools '13, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium (2013), http://dl.acm.org/citation.cfm?id=2512734.2512780
16. Kounev, S., Buchmann, A.: Simqpn—a tool and methodology for analyzing queueing petri net models by means of simulation. Performance Evaluation **63**(4-5), 364–394 (2006)
17. Kunkel, J., Ludwig, T.: Iopm – modeling the i/o path with a functional representation of parallel file system and hardware architecture. In: Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on. pp. 554–561 (Feb 2012)
18. Mitschele-Thiel, A., Müller-Clostermann, B.: Performance engineering of SDL/MSC systems. Comput. Netw. **31**(17), 1801–1815 (1999)
19. Pallis, G., Stamos, K., Vakali, A., Katsaros, D., Sidiropoulos, A.: Replication based on objects load under a content distribution network. In: 22nd International Conference on Data Engineering Workshops (ICDEW'06). pp. 53–53. IEEE (2006)
20. Rygielski, P., Kounev, S., Tran-Gia, P.: Flexible Performance Prediction of Data Center Networks using Automatically Generated Simulation Models. In: Proceedings of the Eighth International Conference on Simulation Tools and Techniques (SIMUTools 2015). pp. 119–128 (August 2015). https://doi.org/10.4108/eai.24-8-2015.2260961
21. Rygielski, P., Kounev, S., Zschaler, S.: Model-Based Throughput Prediction in Data Center Networks. In: Proceedings of the 2nd IEEE International Workshop on Measurements and Networking (M&N 2013). pp. 167–172 (October 2013)
22. Rygielski, P., Seliuchenko, M., Kounev, S.: Modeling and Prediction of Software-Defined Networks Performance using Queueing Petri Nets. In: Proceedings of the Ninth International Conference on Simulation Tools and Techniques (SIMUTools 2016). pp. 66–75 (August 2016), http://dl.acm.org/citation.cfm?id=3021426.3021437
23. Rygielski, P., Seliuchenko, M., Kounev, S., Klymash, M.: Performance analysis of sdn switches with hardware and software flow tables. In: VALUETOOLS (2016)
24. Rygielski, P., Seliuchenko, M., Kounev, S., Klymash, M.: Performance Analysis of SDN Switches with Hardware and Software Flow Tables. In: Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2016) (October 2016)
25. Stoll, J.: SDN-basierte Lastverteilung für Schicht-7 Anfragen (SDN Rechenzentrum Fallstudie). Bachelor Thesis, University of Würzburg, Am Hubland, 97074 Würzburg, Germany (March 2016)
26. group at Sun Microsystems, P.A.E.: Uperf A network performance tool. Online, www.uperf.org, Accessed 28.08.2016 (2012)
27. Team, R.P.: RYU SDN Framework - English Edition Release 1.0. osrg (2014), https://www.amazon.com/RYU-SDN-Framework-English-Edition-ebook/dp/B00IKME2FO, eBook

28. Varga, A.: Omnet++. In: Modeling and tools for network simulation, pp. 35–59. Springer (2010)
29. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D., Maltzahn, C.: Ceph: A scalable, high-performance distributed file system. In: Proceedings of the 7th symposium on Operating systems design and implementation. pp. 307–320. USENIX Association (2006)
30. de Wet, N., Kritzinger, P.: Using UML models for the performance analysis of network systems. Comput. Netw. **49**(5), 627–642 (2005)