Model-Based Throughput Prediction in Data Center Networks

Piotr Rygielski and Samuel Kounev
Institute for Program Structures and Data Organization,
Karlsruhe Institute of Technology (KIT)
76131 Karlsruhe, Germany
Email: {piotr.rygielski, kounev}@kit.edu

Steffen Zschaler
Department of Informatics
King's College London
London, UK
Email: szschaler@acm.org

Abstract—**In this paper, we address the problem of performance analaysis in computer networks. We present a new meta-model designed for the performance modeling of network infrastructures in modern data centers. Instances of our metamodel can be automatically transformed into stochastic simulation models for performance prediction. We evaluate the approach in a case study of a road traffic monitoring system. We compare the performance prediction results against the real system and a benchmark. The presented results show that our approach, despite of introducing many modeling abstractions, delivers predictions with errors less than 32% and correctly detects bottlenecks in the modeled network.**

I. Introduction

The increasing popularity of Cloud Computing has lead to the emergence of large virtualized data centers hosting increasingly complex and dynamic IT systems and services. Due to the common adoption of virtualization technologies, virtualized data centers are becoming increasingly dynamic [1]. Virtual machines, data, and services can be migrated on demand between physical hosts to optimize resource utilization while enforcing service-level agreements. This dynamism of data center infrastructures makes an accurate and timely performance analysis a challenging problem [2].

In our research, we focus on network infrastructures of modern virtualized data centers. Network infrastructures in such environments introduce several new challenges for performance analysis. The growing density of modern virtualized data centers (increased amount of network end-points), the high volume of intra-data-center traffic, or new traffic sources in the management layer of virtualized environments are only some examples of such challenges.

In this paper, we model the performance-relevant aspects of the major common building blocks of modern data center network infrastructures. This includes, for example, network topology, links, nodes, the basic network configuration (e.g., addressing, routing, protocols), the traffic sources and the characteristics of the workloads they generate. These and other basic network elements serve as the basis for building network infrastructures in virtualized data centers.All network virtualization approaches are built on top of physical infrastructure, have configuration, and carry network traffic—thus accurate performance modeling is important at this level.

This paper extends the contributions of our work-in-progress paper [3]. The major contributions of this paper over [3] are: redesigned meta-model; extended model transformations so that simulation can be generated fully automatically; experimental validation of the approach in a case study.

The major contributions of this paper are the following. Firstly, we present the Network Infrastructure (Sub-)meta-model of Descartes Meta-Model (DMM) [4], referred to in short as *DNI meta-model*. Secondly, we provide the model transformation that transform a DNI model (descriptive) to OMNeT++ simulation models (predictive) to enable quantitative evaluation. Thirdly, we present a city traffic monitoring use case and define practical scenarios, which are later modeled with DNI. We conduct experiments and measure the performance of network infrastructures. Finally, we show that the obtained performance predictions reflect the performance of real networks; that confirms that the modeling capabilities of our approach are good in the investigated scenarios.

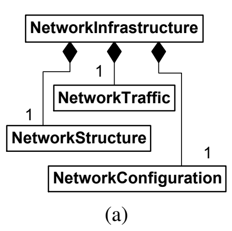
A novel feature of our approach is the generic character of the meta-model. We aim to abstract the highly-detailed specification of network technologies and focus on the modeled system as a whole accepting the possible loss of accuracy. Another benefit is the flexibility in selecting the trade-off between the overhead and accuracy of the analysis (based on the model solving technique): once a system is modeled, multiple transformations to various predictive models may be provided.

The rest of this paper is organized as follows: In Section II, we introduce our approach to performance modeling and prediction. Section III presents the DNI Meta-model and describes its details. Model transformations are described in Section IV. In Section V, we present our traffic monitoring case study, its scenarios, conducted experiments and discuss the results. Finally, we conclude the work in Section VI.

II. Approach

The modeling approach we propose is based on a new meta-model for modeling network infrastructures in virtualized data centers. This meta-model, which we refer to as Descartes Network Infrastructure (DNI) meta-model, is part of our broader work in the context of the *Descartes Meta-Model* (DMM) [4], an architecture-level modeling language for dynamic IT systems and services.

Our approach assumes that instances of the DNI Meta-model are automatically transformed to predictive stochastic



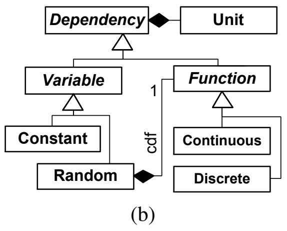


Fig. 1: (a) Root of the meta-model, and (b) dependencies.

models (e.g., product-form queueing networks or stochastic simulation models) by means of model-to-model transformations. Thus, our modeling approach does not require explicit knowledge and expertise in stochastic modeling and analysis. The DNI Meta-model has been designed to support describing the most relevant performance influencing factors that occur in practice while abstracting fine-granular low level protocol details. Our approach is designed to support the implementation of different transformations of the high-level DNI models to underlying predictive stochastic models (by abstracting environment-specific details, transformations to multiple predictive models are possible), thereby providing flexibility in the trade-off between the overhead and accuracy of the analysis. In this paper, we present a single transformation to demonstrate the applicability of the approach.

III. Network Infrastructure meta-model

The DNI meta-model[1] covers three main parts of every data center network infrastructure: structure, traffic and configuration. It is implemented in Ecore using the Eclipse Modeling Framework (EMF). An initial preliminary version of the DNI Meta-model was presented as a work-in-progress paper in [3]; since then, the meta-model has evolved significantly and therefore we present a brief overview of its major parts in the following.

The root element of the DNI Meta-model (`NetworkInfrastructure`) connects the three main parts mentioned formerly: network structure, traffic and configuration (see Fig. 1a). To analyze the performance of any network infrastructure, one must know how the network is physically built (`NetworkStructure`), how it is configured (`NetworkConfiguration`) and how it is used (`NetworkTraffic`). Every numeric value in the model is modeled as a `Dependency` (Fig. 1b). The `Dependency` represents a `Variable` (constant or random) or a `Function`. Additionally, each `Dependency` can be accompanied with a `Unit`. Examples of a `Dependency` can be the following descriptions of a parameter: *"exponentially distributed with mean value of* $100ms$*"*, or just *"5Mbps"*.

A. Network Configuration

The network-configuration meta-model is presented in Figure 2. Currently, the `NetworkConfiguration` contains information about routes, protocols and protocols stacks. In the model, we describe a snapshot of the current routes in the system; we do not explicitly consider dynamic routing as this

[1]For more implementation details, please refer to the auxiliary material that is available on-line under: http://bit.ly/DNI-model

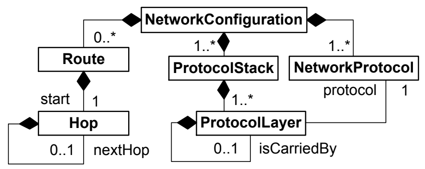


Fig. 2: Meta-model of Network configuration.

would require detailed information about routing algorithms which is abstracted here. In the meta-model, a `Route` consists of `Hops` and each `Hop` references a `NetworkInterface`. The term *route* is an abstraction; we do not store information about dynamic routing protocols. A `ProtocolStack` is an ordered set of `ProtocolLayers`, where each `ProtocolLayer` references a single `NetworkProtocol`. The `NetworkProtocol` itself is described by a generic set of parameters such as, for example, overheads introduces by the data unit headers.

B. Network Structure

The part of the meta-model representing the network structure is depicted in Figure 3. The `NetworkStructure` is a graph consisting of `Nodes` and `Links` connected through `NetworkInterfaces`. All nodes, links and interfaces can be either physical or virtual; each virtual network element is hosted on a `PhysicalNode`. The performance-relevant influencing factors of every element in the `NetworkStructure` are described using `PerformanceSpecification` entities (abbreviated as `PerfSpec` in Fig. 3), both for physical and virtual elements. We distinguish end nodes (e.g., virtual machine, server) and intermediate nodes (e.g., switch, router), because their performance descriptions are different, e.g., end nodes do not utilize information about forwarding performance.

C. Network Traffic

In a data center, most of the network traffic is generated by deployed applications. This includes also the hypervisors (applications) which can trigger, e.g., VM migrations (traffic). As depicted in Figure 4, the DNI meta-model, network traffic is generated by `TrafficSources` that originate from `SoftwareComponents`. Software components are deployed on end nodes. Each `TrafficSource` generates traffic `Flows` that have exactly one source and possibly multiple destinations. The `Flow` destinations are located in `Nodes` and can be uniquely identified by a set of protocol-level addresses. Each `TrafficSource` can generate a specified set of flows. The information about the precise transmission time of a flow is modeled in the workload model (`GenericWorkload`). Each flow can be described by means of various flow descriptions; in this paper, we use a `GenericFlow` description capturing the amount of transferred data. The meta-model can be extended to support other traffic models that can be found in the literature, e.g., [5].

IV. Transformation

An instance of the DNI Meta-model provides a descriptive model of a network. To conduct performance analysis, a given

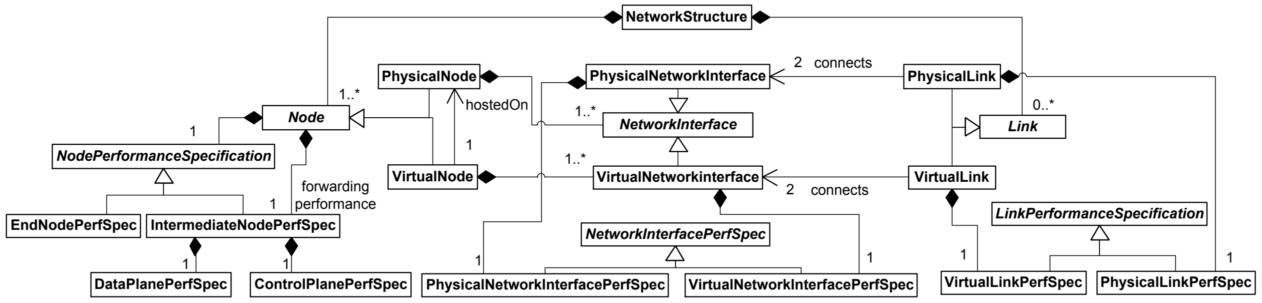


Fig. 3: Meta-model of Network structure.

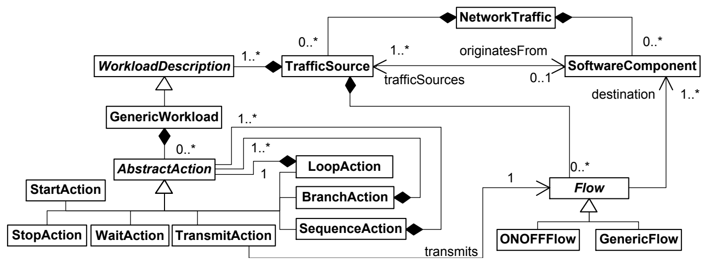


Fig. 4: Meta-model of network traffic

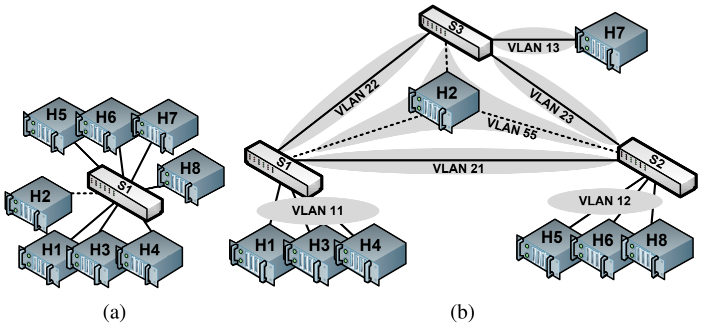


Fig. 5: Experimental environment, network topology and configuration; (a) scenario #1 and #2, (b) scenario #3. Dashed links are used for monitoring, solid links for data traffic.

DNI model must be transformed into a predictive model. In this section, we briefly describe a transformation that transforms an instance of the DNI Meta-model to an OMNeT++ simulation model [6]. The OMNeT model utilizes the INET library, which provides implementations of standard network protocols (e.g., IP, Ethernet, TCP, UDP), thus in current version, only these protocols are supported by the transformation, although the meta model does not have this limitation.

The transformation has been implemented using the Epsilon Framework [7] and is available in the auxiliary material. During an execution of the transformation, entities from the DNI model are automatically translated into entities of the OMNeT model. The meta model of OMNeT simulation has been semi-automatically obtained (using Xtext[8]) from the grammar included in the OMNeT documentation. The transformation consist of rules that define how an input entity should be interpreted by the destination model. "DNI!Link-to-OMNET!Channel", "DNI!Node-to-OMNET!SubModule" are examples of such rules. The resulting OMNeT model is later translated into simulation files using templates defined in model-to-text transformation language EGL (Epsilon Generation Language [7]). The simulation configuration files (e.g., *omnetpp.ini*) are generated from the input DNI model using similar templates. Due to limited space, we cannot provide all details about the transformation. The auxiliary material contains the models, meta models, transformation rules and code generation templates for reference.

The transformation currently assumes that we use standard protocols (L2, L3 and L4 protocols in OMNeT++), although our meta-model allows to model the coarse behavior of any—even custom implemented—network protocol. The matching between the protocols specified in the meta model and the protocols supported by the transformation is done based on the protocol name. We have initially concentrated on these standard protocols as they are the most ubiquitous. According to Shiravi et al. [9], about 97% of current traffic in data centers is carried by IP; additionally, 99% of IP traffic carries TCP or UDP segments.

V. Experimental Validation

To validate our modeling and analysis approach, we have conducted a case study in a data center hosting a traffic monitoring application. We first model the deployed system using the DNI Meta-model and then use the OMNeT++ simulation to predict the network performance. We compare the OMNeT++ results against measurement on the real system as well as against the results obtained by the uperf benchmark [10].

A. Hardware and Configuration

The system under study was deployed in a local data center in an environment consisting of eight servers and three switches. Each server is equipped with an eight-core processor with $3.3GHz$, $16GB$ of memory, and four $1Gbps$ Ethernet ports. The servers are running Ubuntu 12.04-Server. The switches are HP ProCurve 3500yl, each with 24 $1Gbps$ Ethernet ports. The physical topology and the configuration of the network environment is depicted in Figure 5. The host $H2$ connected to the switches using VLAN 55 is used to acquire the monitoring data from switches using SNMP.

B. Case Study

The system under study is a traffic monitoring application based on results from the Transport Information Monitoring Environment (TIME) project [11] at the University of Cambridge. The system consists of multiple distributed components and is based on the SBUS/PIRATES (short SBUS) middleware [12].

Due to technical limitations of the SBUS implementation (single threaded implementation of the SBUS wrapper that wraps each component), we used uperf as a reference for experiments under high load to exclude the influences of software bottlenecks on the network performance. uperf benchmark [10] is a network performance tool that supports modeling and replay of various network traffic patterns. uperf was shown to emulate the network traffic without exhibiting any scalability or stability issues under high load.

In the case study, we consider two kinds of components: cameras and license plate recognition (LPR) components. The cameras are located in the city and take pictures of cars that are speeding or entering a paid zone. Each camera is connected to a local SBUS component that sends the picture together with a time stamp to the LPR components. The LPR components are deployed in a data center due to their high consumption of computing resources. LPRs receive the pictures emitted by cameras and run a recognition algorithm to identify the license plate numbers of the observed vehicles.

In our experiments, we assume that there are N cameras connected with the data center using a dedicated network line. The network line is assumed to be a leased channel that is characterized with a maximum bandwidth X. The channel is only used by the cameras—there is no other traffic in the network. Every camera sends pictures of size L every p units of time. Each picture is transmitted with additional parameters like, for example, time stamp, location identifier and measured speed of the car. We model the intensity of the road traffic with λ photographed cars per second. We consider the following scenarios.

Scenario #1: A single camera is connected with the data center using a dedicated leased line. The line has maximum theoretical bandwidth of $X = 1Gbps$. The road traffic intensity is $\lambda = 2$ photographed cars per second. How much bandwidth will be utilized by the system on the path between LPR and the switch if the traffic intensity increases? In this scenario, the camera and the LPR is deployed in a single VLAN with servers connected in a star topology (Figure 5a).

Scenario #2: Given an area in the city with a single network end-point, we consider to add new cameras to the area that is connected using a leased line with bandwidth $X = 1Gbps$. How many cameras can be handled without congestion by this connection for a given road traffic intensity? In this scenario, all cameras are deployed in a single VLAN with servers connected in a star topology (Figure 5a). The cameras transmit the data over a single shared network link to the LPR component.

Scenario #3: Given multiple areas in the city connected to the data center, we consider the situation with multiple cameras and LPR components deployed on different servers due to predefined distribution of computational workload. The cameras deployed in VLAN 11 transmit the data to the LPRs deployed in VLAN 12. Additionally, we model other source of external traffic by deploying additional camera component in VLAN 13 (see Figure 5b). In this scenario, we examine the bottlenecks in VLAN 21 and the switches $S1$ and $S2$ (referred to as scenario 3A). Moreover, we study additional changes in topology and configuration in the following sub-scenarios. In the scenario 3B, we study the changes in the network performance if the connection between switch $S1$ and $S3$ (VLAN 22) fails. Scenario 3C assumes the reaction of the network operator where the camera hosted on $H7$ is disabled. Finally, scenario 3D represents the situation where VLAN 22 is brought back to operation, however the camera component on host $H7$ remains turned off.

C. Measurements

In every scenario, we measure the amount of the traffic flowing through the network interfaces of all switches. We use the set of counters located in the switches to measure the number of bytes transmitted through each interface. We read the values of the counters through SNMP every second and calculate the average throughput for that interval. Host $H2$ takes the measurements using the isolated VLAN 55 (no traffic can be routed to or from that VLAN). The sizes of transmitted messages were constant in all experiments. The experimental network was isolated from other networks (e.g., Internet). During the measurements, all think times were modeled as exponentially distributed; confidence intervals are calculated for significance $\alpha = 0.05$.

In every experiment we send predefined amount of pictures (usually 10 000 for each camera) and execute the experiment 30 times for SBUS and 16 times for OMNeT. We use 16 repetitions for OMNeT due to long simulation time; to simulate one real second, the generated OMNeT model needs about 100 seconds.

D. Results

In the two first scenarios, we have examined mainly the correctness of the modeling approach and the transformation.

Scenario #1 In the first scenario, we set the message size to $2000kB$ and vary the think time p for a single sender. In experiment A (Figure 6 left), we set the think time to $500ms$ and decreased it in steps of $50ms$. The measured throughput grow exponentially for lower think times. Note, that the throughput values for SBUS drop when decreasing the p below $100ms$. This phenomenon is caused by scalability problems of SBUS (single-threaded implementation of the SBUS wrapper). This situation was also observed in the second scenario.

Both, OMNeT and uperf followed the trend of SBUS, slightly overestimating the actual throughput. Additionally, we investigated a range of smaller think times in experiment B (Figure 6 right). In this experiment, the prediction errors were lower than 20%.

Scenario #2 In the second scenario, we varied the think times and the number of senders. Due to the scalability issues of SBUS, the measurements were taken only for OMNeT and uperf (uperf served as reference). The results are depicted

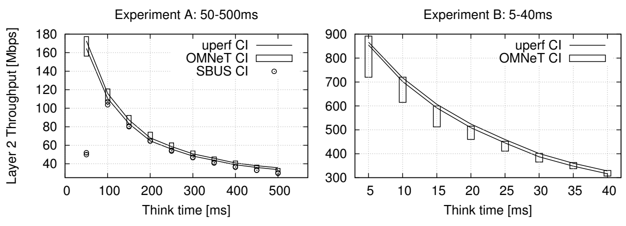


Fig. 6: Scenario #1: Confidence intervals for the mean throughput for think time 50–500ms (left) and for 5–40ms (right).

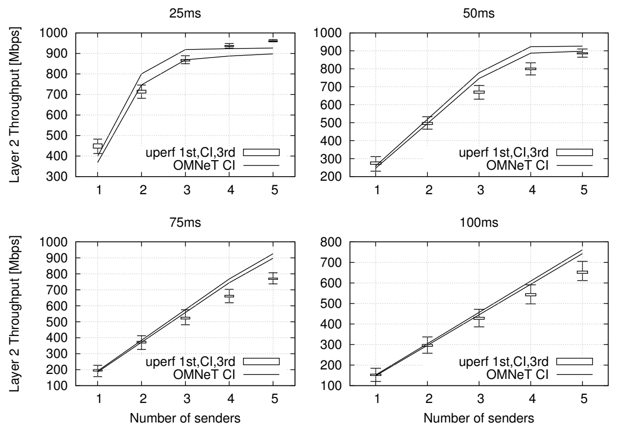


Fig. 7: Scenario #2: Confidence intervals for varying number of senders and think time p. For uperf, 1st and 3rd quartiles are shown additionally.

in Figure 7; the relative errors are given in Table I. For think time $p = 75ms$ and $100ms$, we see linear dependency between the number of senders and the achieved throughput. For $50ms$, the maximum throughput was reached with five senders. The situation was similar for the $25ms$ run. In the presented experiments, the relative prediction error was higher than in scenario #1; the extremes of confidence intervals of both models were a maximum 21% of the reference value apart. According to the uperf model, the maximum achievable bandwidth is $963Mbps$; OMNeT reports the maximum as $935Mbps$ (underestimating by 3%).

In the next scenario, we deploy multiple cameras and LPRs; there are multiple traffic sources and destinations. Moreover, we decrease the message size to $L = 400kB$ to address the scalability issues of the SBUS. Here, the validation is carried against SBUS to increase the reality of the configuration.

TABLE I: Prediction errors and goodness of fit for scenarios #1 and #2.

Scenario #1	A: OMNeT vs. PIRATES	A: uperf vs. PIRATES	A: OMNeT vs. uperf	B: OMNeT vs. uperf
Error (min–max)	$0 - 14.1\%$	$1 - 16.4\%$	$0 - 11.6\%$	$0 - 15.8\%$
Fit R^2	0.9877	0.9859	0.9885	0.99743
Scenario #2	$25ms$	$50ms$	$75ms$	$100ms$
Error (min–max)	$0 - 20.2\%$	$0 - 17.1\%$	$0 - 20.9\%$	$0 - 17.5\%$
Fit R^2	0.9287	0.9793	0.9963	0.9980

Scenario #3 In the third scenario, we deployed the following set of traffic sources and destinations: $H1 \rightarrow H7$, $H3 \rightarrow H6$, $H4 \rightarrow H5$, $H5 \rightarrow H6$, and in the sub-scenarios #3A and #3B, $H8 \rightarrow H7$. The components were configured to spawn two threads, each sending a picture of size $L = 400kB$ every $1ms$ so that the network is fully loaded. There were also multiple SBUS wrapper instances running in the background to exclude the influence of the SBUS scalability issues on the network. In every run, each host sent 10000 pictures. We stopped the measurements once all cameras reported successful transmission of all pictures. The measured values of throughput are presented in Table II. Prediction errors were calculated as the difference between the mid points of confidence intervals.

In the scenario #3A, our model kept the prediction error at acceptable level not exceeding 25%. In most cases, the prediction was an overestimation, however three results are underestimated. All existing bottlenecks were detected but there was also one false positive: the bandwidth on path $S3 \rightarrow H7$ was overestimated by 24% and a bottleneck was reported, although there were still free resources on that link.

In the scenario #3B, there was only one bottleneck, which was detected successfully. Due to the failure of VLAN 22, there was an overload situation on the only link connecting VLAN 11 with VLANs 12 and 13. Most of the prediction errors were under 20% except for two overestimations—by 32% and 26%.

In the scenario #3C, we lowered the amount of traffic by disabling one camera, however it did not affect the bottleneck between switches $S1$ and $S2$. OMNeT discovered a bottleneck on the path $S2 \rightarrow H8$ but there were still free resources in reality. Although the bottleneck $S1 \rightarrow S2$ visibly slowed down the other flows, the prediction errors remained at a similar level—up to 27%.

Scenario #3D represents the situation where the VLAN 22 is repaired but the camera on $H8$ remains off. In this case, the load in the network was more balanced as eight out of 12 links reached saturation of over 85%. The predictive model discovered all eight highly saturated links and reported all possible bottlenecks correctly. Prediction errors remained under 30%.

Summarizing, the automatically generated simulation model in OMNeT provided accurate predictions not exceeding the average prediction error of 32% in the average worst case. The model correctly recognized all bottlenecks and provided only two false positives. Additionally, in case of high traffic load (bottlenecks), the relative prediction errors were low and did not exceed 13% (up to $110Mbps$); in case of false positives the error was higher—up to 25%. We stress that the presented predictions were obtained by an automatically generated simulator; the simulation model was generated from a DNI model where most of low-level details were abstracted.

VI. Conclusion and Future Work

In this paper, we presented a new meta-model for performance modeling of data center network infrastructures. The meta-model was designed to address the challenges of network performance analysis in modern data centers by abstracting too detailed and technology-specific information, while covering

TABLE II: Measured and predicted bandwidth between network nodes. Confidence intervals are given in Mbps. Bottlenecks are marked in bold.

Link	SBUS		OMNeT		Relative
Source $\rightarrow$	*Mbps*		*Mbps*		error %
Destination	lCI	uCI	lCI	uCI	
Scenario #3A: VLAN 22, $H8 \rightarrow H7$					
H1→S1	352	413	440	448	13.4%
H3→S1	396	499	407	494	0.1%
H4→S1	540	563	595	611	9.1%
H5→S2	533	545	667	684	25.4%
H7→S3	567	795	644	654	−9.5%
H8→S2	376	440	491	500	18.7%
S1→S2	**915**	**932**	**926**	**944**	1.3%
S1→S3	352	413	440	448	13.4%
S2→H5	616	638	667	684	7.6%
S2→H6	**860**	**872**	**926**	**944**	8.1%
S2→H8	568	795	591	600	−17%
S2→S3	376	441	491	500	18.5%
S3→H7	680	803	**931**	**948**	23.7%
S3→S2	566	793	591	600	−16.8%
Scenario #3B: no VLAN 22, $H8 \rightarrow H7$					
H1→S1	210	246	298	314	31.9%
H3→S1	277	302	303	323	7.7%
H4→S1	402	440	355	367	−15.1%
H5→S2	443	533	364	474	−13.1%
H7→S3	282	444	349	446	6.1%
H8→S2	470	530	630	647	25.8%
S1→S2	**897**	**946**	**937**	**950**	1.7%
S2→H5	410	469	393	418	−8.8%
S2→H6	732	851	591	680	−19.8%
S2→H8	289	457	349	446	3.2%
S2→S3	695	822	623	712	−12.5%
S3→H7	696	822	623	712	−12.5%
S3→S2	243	445	349	446	9.5%
Scenario #3C: no VLAN 22, $H8 \nrightarrow H7$					
H1→S1	244	291	311	323	15.9%
H3→S1	272	287	295	312	8.6%
H4→S1	366	394	346	356	−8.3%
H5→S2	443	521	347	457	−15.1%
H7→S3	218	410	300	427	11.2%
S1→S2	**931**	**952**	**945**	**951**	0.4%
S2→H5	368	409	383	405	0.6%
S2→H6	714	807	584	676	−16.8%
S2→H8	545	792	**872**	**889**	24.5%
S2→S3	247	302	349	363	26.3%
S3→H7	248	306	349	363	25%
S3→S2	219	412	300	427	10.6%
Scenario #3D: VLAN 22, $H8 \nrightarrow H7$					
H1→S1	**819**	**846**	**807**	**822**	−2.4%
H3→S1	354	365	410	495	29.2%
H4→S1	540	524	604	612	15.1%
H5→S2	527	548	677	685	26.1%
H7→S3	**929**	**941**	**900**	**916**	−2.8%
S1→S2	**884**	**902**	**938**	**947**	5.4%
S1→S3	**823**	**849**	**807**	**822**	−2.8%
S2→H5	546	571	677	685	21.3%
S2→H6	**864**	**880**	**938**	**947**	7.9%
S2→H8	**930**	**939**	**804**	**818**	−13.1%
S3→H7	**824**	**849**	**904**	**919**	8.7%
S3→S2	**926**	**940**	**804**	**818**	−13%

important performance-relevant aspects of network infrastructures. We showed how DNI models can be automatically transformed to OMNeT simulation models to obtain quantitative evaluation of networks performance. The obtained simulation models were validated in the traffic monitoring case study. The conducted experiments showed that the presented transformation correctly transforms the DNI model into simulation. In simple scenarios our approach delivers accurate predictions with maximal error of 21% and the model fits the measured data with $R^2 > 0.928$.

The third scenario shows how the model predicts the throughput in more complex situations. The experiments conducted in the third scenario showed that the generated model reflects real performance with maximal error of 32%. Additionally, the simulation model recognized all real bottlenecks providing only two false positives. This is a satisfactory prediction accuracy given the high degree of abstraction in our meta model.

In the future, we will focus on extending the meta model and on developing further transformations. We plan to provide additional modeling abstractions to cover virtualization aspects of modern data center networks [13]. The experiments conducted in this work are also an incentive to provide transformation to other simulation models, so that performance predictions are obtained quicker and the prediction accuracy remains at a satisfactory level or may be improved.

This work is a part of RELATE project supported by the European Commission (Grant no. 264840ITN).

References

[1] K. Juszczyszyn, P. Światek, P. Stelmach, and A. Grzech, "A configurable servicebased framework for composition, delivery and evaluation of composite web services in distributed QoSaware ICT environment," *International Journal of Cloud Computing*, vol. 2, no. 2, pp. 258–272, 2013.

[2] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments," in *Proc. of the 1st Int. Conf. on Cloud Computing and Services Science*, 2011, pp. 563–573.

[3] P. Rygielski, S. Zschaler, and S. Kounev, "A Meta-Model for Performance Modeling of Dynamic Virtualized Network Infrastructures," in *Proc. of the 4th ACM/SPEC Int. Conf. on Performance Engineering (ICPE'13).* New York, NY, USA: ACM, April 2013, pp. 327–330, Work-In-Progress Paper.

[4] F. Brosig, N. Huber, and S. Kounev, "Architecture-Level Software Performance Abstractions for Online Performance Prediction," *Elsevier Science of Computer Programming Journal (SciCo)*, 2013.

[5] A. J. Field, U. Harder, and P. G. Harrison, "Network Traffic Behaviour in Switched Ethernet Systems," in *MASCOTS 2002, 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, October 2002, pp. 32–42.

[6] A. Varga, "The OMNeT++ discrete event simulation system," in *Proc. of the European Simulation Multi-conference*, 2001, pp. 319–324.

[7] D. Kolovos, R. Paige, and F. A. Polack, "The Epsilon Transformation Language," in *Theory and Practice of Model Transformations, vol. 5063 of LNCS.* Springer, 2008, pp. 46–60.

[8] "Xtext, Language Development Made Easy," Online, 2013. [Online]. Available: http://www.eclipse.org/Xtext

[9] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[10] "uperf A network performance tool," Performance Applications Engineering group at Sun Microsystems, 2012.

[11] J. Bacon, A. Beresford, D. Evans, D. Ingram, N. Trigoni, A. Guitton, and A. Skordylis, "TIME: An open platform for capturing, processing and delivering transport-related data," in *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas*, 2008.

[12] D. Ingram, "PIRATES Data Representation," http://www.cl.cam.ac.uk/research/time/pirates/docs/datarepr.pdf, 2009.

[13] P. Rygielski and S. Kounev, "Network Virtualization for QoS-Aware Resource Management in Cloud Data Centers: A Survey," *PIK — Praxis der Informationsverarbeitung und Kommunikation*, vol. 36, no. 1, pp. 55–64, 2013.