Data Center Network Throughput Analysis using Queueing Petri Nets

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

***Abstract*—In this paper, we contribute performance modeling and analysis approach in computer networks. We present a meta-model designed for the performance modeling of network infrastructures in modern data centers. Instances of our meta-model can be automatically transformed into stochastic simulation models for performance prediction. In this paper, we present a transformation to Queueing Petri Nets (QPNs). We show that despite the high level of abstraction of the QPN models, we are able to predict the utilization of resources with good accuracy within a short time.**

I. Introduction

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

In our research, we focus on network infrastructures of modern virtualized data centers—that is, networks that are operated by a single administrative unit. Network infrastructures in such environments introduce several new challenges for performance analysis. The growing density of modern virtualized data centers (increasing amount of network end-points), the high volume of intra-data-center traffic, or the new traffic sources introduced in the management layer of virtualized environments, are some examples of such challenges.

In this paper, we model the performance-relevant aspects of the common parts of modern data center network infrastructures. This includes, among others, network topology, links, nodes, routes, protocols as well as 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. Modern network virtualization approaches rely on these building elements—thus appropriate performance modeling at this level is important.

We propose a generic approach to network performance prediction. We present the Descartes Network Infrastructure (DNI) meta-model as a part of the Descartes Modeling Language (DML) [2]. Network models built using our domain-specific modeling language (DNI meta-model) can be automatically transformed to various predictive models. Thus, our approach requires a single input DNI model and offers multiple predictive models without requiring expertise in each of them.

The main contributions of this work are the following: We propose a generic approach to network performance prediction that supports multiple performance analysis techniques based on models at different levels of abstraction and accuracy. We contribute a model transformation that translates a DNI model into a Queueing Petri Net (QPN) model that can be simulated with the SimQPN simulator [3]. Finally, we validate the transformation by conducting experiments to evaluate the prediction accuracy and simulation time.

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 supported model solving techniques): once a system is modeled, multiple transformations to predictive models of different granularity are provided.

The rest of this paper is organized as follows: In Section II, we briefly review similar model-based performance prediction approaches. In Section III, we introduce our approach to performance modeling and prediction. Section IV briefly presents the DNI meta-model. The transformation to the QPN formalism is described in Section V. In Section VI, we present our validation of the approach and discuss the evaluation results. Finally, we conclude the work in Section VII.

II. Related Work

There is a large body of existing work on performance modeling of communication networks, for example [4], [5]. Existing modeling approaches are mostly based on stochastic models such as classical product-form queueing networks, extended or layered queueing networks, stochastic Petri nets, stochastic simulation models, and so on. Building such models requires experience and expertise in stochastic modeling and analysis, which network administrators typically do not have. Moreover, such models usually capture only selected specific aspects of the network infrastructure whereas other aspects are abstracted. To address this issue, the research community has proposed high-level network modeling approaches that support automatic generation of low-level predictive models.

Most existing model-based approaches (e.g., [6], [5], [7]) are based either on black-box statistical models or on highly-detailed protocol-level simulation models (e.g., [8], [9], [10]). The former do not consider the internal network structure and topology while the latter focus only on selected parts of the network infrastructure and do not capture the link to the running applications and services that generate the traffic. Other approaches model the network environment without providing support for performance analysis, e.g., [11], [12].

Regarding the stochastic Petri nets, such as QPNs, and their usage in a networking context, so far they have mainly been used to model specific network components at a high level of detail, for example, [13], [14], [15]. Additionally, in other works (e.g., [16], [17]), networks appear as a part of a larger modeling landscape and are typically modeled as a black-box.

In summary, most approaches to performance analysis of communication networks concentrate on studying specific networking technologies that are modeled at the protocol level typically using low level simulation models or highly-detailed specification languages. Such models are highly specific and are difficult to use in a different system configurations than the one for which they have been designed. On the other hand, existing measurement-based or black-box modeling approaches are coarse-grained and do not provide means to capture the effects of the individual performance-influencing factors. We bridge this gap by providing the trade-off between the two extremes and allowing flexible performance analysis at the required abstraction level.

III. Approach

The modeling approach we propose is based on a meta-model for modeling network infrastructures in virtualized data centers. This metamodel, which we refer to as Descartes Network Infrastructure (DNI) metamodel, is part of our broader work in the context of the *Descartes Modeling Language* (DML) [2], an architecture-level modeling language for modeling Quality-of-Service and resource management related aspects of modern dynamic IT systems, infrastructures and services. The DNI metamodel 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 assumes that instances of the DNI metamodel are automatically transformed to predictive stochastic 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. Our approach is designed to support the implementation of different transformations of the descriptive 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 transformation that translates DNI models to QPNs. The QPN models are later simulated using the SimQPN simulator [3], [18]. In the next section, we briefly describe the key parts of the DNI metamodel and in Section V, we provide details about the transformation itself.

IV. Network Infrastructure Meta-model

The DNI meta-model covers three main parts of every data center network infrastructure: structure, traffic and configuration. The DNI meta-model is described in more details in [19], [20] and is available online[1]. In the next paragraphs, we briefly characterize the main parts of the meta-model.

The first part of the DNI meta-model—network structure—is intended to model the topology of the network. The meta-model contains entities such as nodes and links connected through network interfaces. All nodes, links and interfaces can either be physical or virtual; each virtual network element is hosted on a physical node. We describe the performance-relevant parameters of every element in the model. We distinguish end nodes (e.g., virtual machine, server) and intermediate nodes (e.g., switch, router), because their performance descriptions are different.

The configuration of a network contains information about routes, protocols and protocols stacks. We use this information to calculate the paths in the topology graph and to coarsely estimate the overheads introduced by the protocols. In the model, we describe a snapshot of the current routes in the system, disregarding if the system uses static or dynamic routing. In the meta-model, a route consists of a list of references to network interfaces. The protocols are described by a set of generic parameters such as, for example, overheads introduced by the data unit headers.

One cannot analyze network performance without considering the traffic. In a data center, most of the network traffic is generated by deployed applications. This includes also the hypervisors which can trigger, e.g., VM migrations.

In the DNI meta-model, network traffic is generated by traffic sources that are deployed on end nodes. Each traffic source 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. Flows can be composed in a workload model that defines how each flow is generated (e.g., with sequences, loops, or branches). In this paper, we describe a flow by specifying the amount of transferred data. The meta-model can be extended to support other flow descriptions that can be found in the literature, for example, [21].

V. Transformation

An instance of the DNI meta model is a descriptive model of a network. To conduct performance analysis, the DNI model must be transformed into a predictive model. In this section, we describe a transformation that transforms an instance of the DNI meta model to a QPN model that can be simulated using the SimQPN simulator, which is part of QPME (Queueing Petri Net Modeling Environment) [3].

The QPN formalism was introduced by Bause in [22]. The graphical notation used in this section is summarized in Figure 1.

[1]Auxiliary material on-line: http://bit.ly/DNI-model

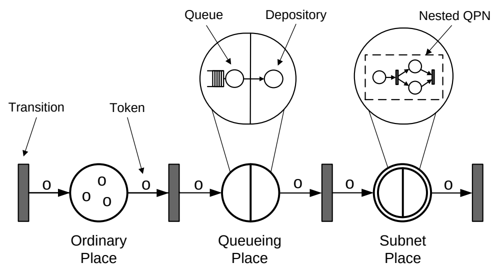

Fig. 1: Notation used in QPN diagrams. Excerpted from [17].

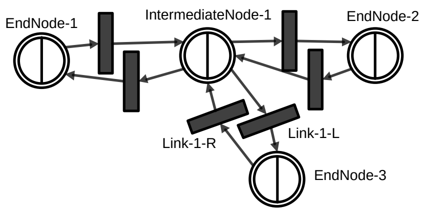

Fig. 2: QPN representation of network nodes and links.

A. Top-level Network

The transformation begins with translation of all network nodes into subnet places. For each *Node*, a subnet is created. To reflect the topology, the subnets are connected with links that are represented by two transitions—each for one of both directions. The incidence functions for links are defined as non-blocking, that is, they fire immediately for a single token of any color and deposit the token in the succeeding place in the graph. As an example, a QPN representing three end nodes connected with a single intermediate node is depicted in Figure 2.

B. End Node

Each *EndNode* represents a machine that can host a software stack that may contain multiple traffic sources. An end node without traffic sources specified in the input DNI model is transformed to an intermediate node or removed. The structure of the subnet representing an end node is depicted in Figure 3 and is described as follows. All the tokens that are directed to an end node subnet are placed in the input place. Next, they are forwarded to the queueing places that represent receive queues of the network interfaces (*eth-rx*). The *input-trans* transition directs the tokes to the proper interface based on the routing information contained in the DNI model (the internal structure of the *input-trans* and other transitions is abstracted in Figure 3). The tokens wait in the *eth-rx* queueing places to be later destroyed in the transition named *rx-to-sw*. The destruction of the incoming tokens allows modeling the open workflows—incoming tokens do not interfere with the token generation process. The connections between the *rx-to-sw* transition and the traffic sources are kept only to guarantee the liveness of the QPN—no tokens are transmitted there.

Each traffic source deployed on the given node is represented as a subnet where tokens representing network traffic are generated. The process of transmitting the generated tokens is similar—a transition named *sw-to-tx* selects the appropriate transmit queue of the network interface *etx-tx*, tokens wait in

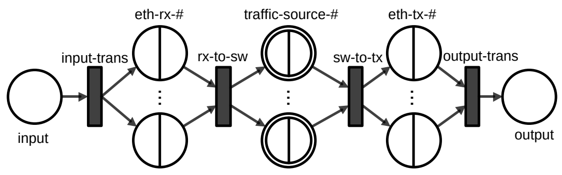

Fig. 3: QPN representation of an EndNode.

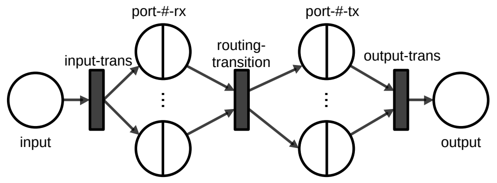

Fig. 4: QPN representation of an IntermediateNode.

the queue and are passed to the output place. After a token is deposited in the output place, the transition that represents a respective link (e.g., *Link-1-R* in Fig. 2) is immediately fired passing the token to the next node according to the routes defined in the DNI model.

C. Intermediate Node

Each *IntermediateNode* represents a forwarding device (e.g., a switch or a router). Its QPN representation is a subnet place with the structure that is similar to the end node (see Fig. 4). The main difference between the end node subnet and the intermediate node subnet is the lack of traffic sources in the latter. The *rx-to-sw* transition, traffic sources subnets, and the *sw-to-tx* transition are substituted with a single *routing-transition*. The incidence function of the *routing-transition* is defined in such a way that tokens incoming from defined receive ports (*port-rx*) are forwarded to proper transmit ports (*port-rx*). The function is defined uniquely, that is, for each receive port and for each token color there is maximally one transmit port where the token should be directed to. In this subnet, contention takes place only in queues of the ports.

D. Traffic Source

A *TrafficSource* is represented as a subnet place in the QPN model. The main responsibility of the traffic source subnet is the generation of tokens according to the workload defined in the DNI model. Figure 5 depicts a QPN model of an exemplary traffic source. All unnamed transitions are generated as mandatory connections between two consecutive places; these transitions are passing all token colors in a non-blocking fashion.

To control the order of actions executed in the workflow the *workload-execution* (WE for short) is defined. The *workload-control* place contains initial marking (tokens at the start of the simulation) of a single WE token. Later, the WE token traverses the places and transitions as defined in the DNI model. The DNI metamodel uses actions for representing workflow elements: start action, wait action, transmit action, stop action, sequence, loop, fork/branch actions. The transformation of the workload is the most challenging part because actions can be nested and called recurrently.

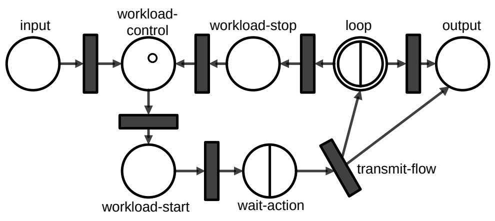

Fig. 5: QPN representation of the TrafficSource.

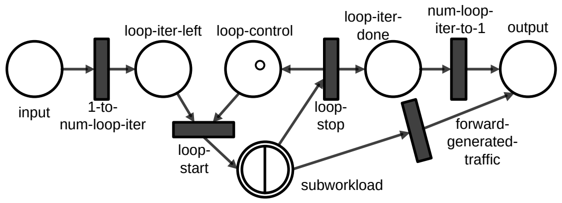

Fig. 6: QPN representation of the workload loop action.

The traffic generation procedure runs as follows. All tokens in the input place are ignored and destroyed in the proceeding transition (due to the open workflow model). The WE token is passed to the *workload-start* place that stems from the *StartAction* defined in the DNI workload model. Each *WaitAction* is translated to a queueing place. A *TransmitAction* is transformed to *transmit-flow* transition. A *transmit-flow* transition passes WE token and produces a token representing a flow. That flow token is immediately deposited in the output place and is ready to be routed and transmitted by the end node. The WE token is passed further to the next action in the workflow (loop in this example) until it reaches the *workload-stop* place. Then, the whole process is repeated.

The *LoopAction* is represented as a subnet and its internal structure is depicted in Figure 6. The single WE token arriving to the input is later transformed into multiple tokens by the *1-to-num-loop-iter* transition. The transition produces exactly the amount of tokens that corresponds to the number of loop iterations defined in the DNI model. The next transition—*loop-start*—is the synchronization point; it requires one token from the *loop-iter-left* and another one from the *loop-control* place. The latter has a single WE token set as initial marking so that *subworkload* can start as soon as the subnet receives a single WE token from outside. When all actions of the *subworkload* are finished, the *loop-stop* transition duplicates the WE token and passes one to the *loop-iter-done* and the second one to the *loop-control* place. Now the next loop iteration can begin, as long as there are WE tokens in the *loop-iter-left* place left. When the *loop-iter-done* place contains the amount of tokens equal to the number of iterations, the transition fires and deposits a single WE token to the output place. The execution of all loop iterations is finished.

The actions that are looped are represented as a subnet for brevity. The internal structure of the *subworkload* subnets corresponds to the traffic source subnet and can contain multiple nested loops or branches. The tokens that represent traffic flows are directly sent to the output place to be immediately passed to the *sw-to-tx* transition on the respective end node level.

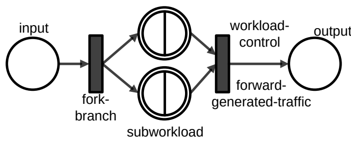

Fig. 7: QPN representation of the workload fork and branch actions.

The structure representing a branch/fork action is depicted in Figure 7. The only difference between fork and branch action is the incidence function in the *fork-branch* transition—for fork it generates tokens on all output connections, for branch only on a single one (selected randomly based on probabilities encoded in the DNI model).

E. Colors Management

The QPME editor and the SimQPN simulator divide tokens into classes called colors. The tokens having the same color are indistinguishable. Thus, all important components of the simulated traffic (e.g., flows) must be modeled as separate colors. To the colors present in every simulation we account WE tokens—one color for each traffic source. Other colors are automatically generated based on the traffic model. We generate a separate color for each flow in the model. Additionally, if there are multiple flows that have the same source and destination but different data sizes, we generate separate colors for these flows.

F. QPN Configuration

The generated QPN is setup to reflect the real contention and synchronization points. Each queue is configured with the FCFS policy and the service times are calculated based on the data size and protocols overheads for each network interface and a color representing a flow. Moreover, each queue has constant predefined maximum capacity; if the capacity is exceeded, the incoming tokens are destroyed (drop-tail queue). SimQPN parameters are set to match the reality so all transitions have equal priority, all places have normal departure discipline, that is, any succeeding transition may fire as soon as all required tokens are available disregarding in which order they arrived.

VI. Validation

To validate the transformation, we conduct a case study in a data center hosting a traffic monitoring application. We first model the deployed system using the DNI metamodel and then use the SimQPN simulation to predict the network performance. We compare the SimQPN predictions against measurements on the real system obtained by running the uperf benchmark [23].

A. Hardware and Configuration

The system under study was deployed in a local data center in an environment consisting of seven servers and three switches. Each server is equipped with four $1Gbps$ Ethernet ports. 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 8. The host $H2$ is used to acquire the monitoring data from switches using SNMP.

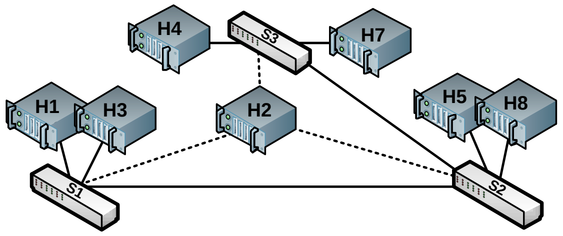

Fig. 8: Network topology used in the experiment. Dashed links are used for monitoring, solid links for data traffic.

B. Case Study

The system under study is a traffic monitoring application based on results from the Transport Information Monitoring Environment (TIME) project [24] at the University of Cambridge. The system consists of multiple distributed components and is based on the SBUS/PIRATES (short SBUS) middleware.

In the case study, we consider two kinds of components: cameras and license plate recognition (LPR). 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 an SBUS component that sends the picture together with a time stamp to the predefined 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.

C. Experiment

In this paper, we validate the transformation using a setup where multiple cameras transmit images using UDP to LPRs deployed on nodes $H5$ and $H8$. We consider three scenarios. In scenario A, two cameras deployed on $H1$ and $H3$ are transmitting images to a single LPR deployed on $H5$. In scenario B, we assume that two new cameras ($H4$ and $H7$) are added to the system. Scenario C represents a situation where an additional LPR was deployed on node $H8$ to provide additional image processing resources.

We investigate the following questions. Firstly, how good the obtained QPN model reflect the real performance behavior assuming non-reliable UDP-like communication. And secondly, the performance of the SimQPN simulation.

D. Measurements

In every scenario, we measure the amount of the traffic flowing through the network interfaces of all switches. We use the 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 an isolated VLAN. The sizes of transmitted messages were constant in all experiments ($800KB$). The experimental network was isolated from other networks (e.g., the Internet). During the measurements, all intergeneration times were modeled as exponentially distributed; confidence intervals are calculated for significance $\alpha = 0.05$. In every experiment we send predefined amount of pictures (3 000 for each camera) and execute the experiment 30 times for uperf and once for QPN (SimQPN cares internally about the required amount of repetitions).

TABLE I: Measured and predicted throughput between network nodes.

Link	uperf (measured)		SimQPN (predicted)	Relative error %
Source → Destination	*Mbps* lCI	uCI	*Mbps* average	
Scenario A ($10ms$): $(H1, H3) \rightarrow H5$				
H1→S1	374	396	400	3.9%
H3→S1	375	396	400	3.8%
S1→S2	605	657	800	26.8%
S2→H5	605	657	800	26.8%
Scenario B ($10ms$): $(H1, H3, H4, H7) \rightarrow H5$				
H1→S1	380	392	400	3.6%
H3→S1	380	392	400	3.6%
S1→S2	661	689	800	18.5%
H4→S3	380	391	400	3.8%
H7→S3	380	392	400	3.6%
S3→S2	621	650	800	25.9%
S2→H5	859	889	978	11.9%
Scenario C ($10ms$): $(H1, H3) \rightarrow H5, (H4, H7) \rightarrow H8$				
H1→S1	388	396	400	2.0%
H3→S1	388	396	400	2.0%
S1→S2	681	704	800	15.5%
H4→S3	388	396	400	2.0%
H7→S3	388	396	400	2.0%
S3→S2	642	667	800	22.2%
S2→H5	681	704	800	15.5%
S2→H8	642	667	800	22.2%

E. Results

The results of the described scenarios are gathered in the Table I. We present throughputs for the most loaded links in the system for a selected intergeneration time (one picture every $10ms$). The throughputs for uperf are given as confidence intervals and for SimQPN as averages (SimQPN does not provide confidence intervals for throughput).

The values reported by SimQPN were close to the real measurements and stable. The stability was expected regarding the introduced abstractions. For underutilized links, the prediction errors were lower than 5%, whereas for heavy utilized links, the errors grew up to 26%. Under heavy load, SimQPN was overestimating the throughput.

We examined additionally the scenario C for various traffic intensities. The results for the connection $S2 \rightarrow H5$ are presented in Figure 9. This chart confirms the overestimations introduced by SimQPN, however, the general trend is followed correctly. For lower link utilizations the predictions are accurate; for higher link utilizations, the errors are up to $200Mbps$ out of 1000 in the worst case. This is a satisfactory prediction accuracy given the high degree of abstraction in our metamodel. Moreover, the presented predictions were obtained by an automatically generated simulation model.

The simulation takes about 7 wall clock seconds on average (measured for scenario C, intergeneration time $10ms$), including loading the input file and storing the results to the disk. Larger simulations take: $30s$ for 30 nodes, $46s$ for 40 nodes, $65s$ for 50 nodes, and $325s$ for 100 nodes. The used version

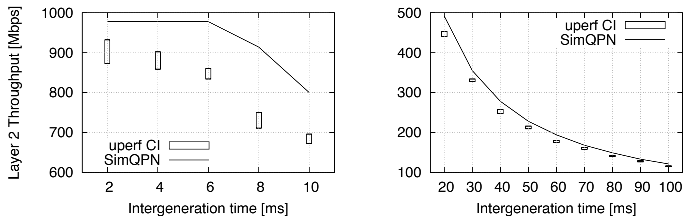

Fig. 9: Throughput on link $S2 \rightarrow H5$ in scenario C for variable intergeneration times.

of SimQPN utilizes only a single CPU core for simulation, however, a parallel version is currently under development and will be released soon.

VII. Conclusions and Future Works

In this paper, we presented a generic model-based approach to network performance prediction. We introduced the *DNI meta-model* and described the transformation to QPNs. In the validation, we showed that the transformation works correctly, the prediction accuracy is acceptable, and the simulation time is short. We stress that the presented predictions were obtained through an automatically generated simulation model from a high-level descriptive model where most low-level details were abstracted.

As part of our future work, we intend to provide an extensive comparison of the generated QPN models against other automatically generated predictive models (e.g., OMNeT++ presented in [20]). Moreover, we aim to provide more transformations to models with different granularities to enable flexibility in performance prediction, so that simulation at different level of detail can be used depending on the required accuracy and time constraints. Finally, we plan to validate our automatically generated models in larger data center conducting multiple case studies and considering further performance metrics.

Acknowledgment

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

References

[1] 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.

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

[3] S. Spinner, S. Kounev, and P. Meier, "Stochastic Modeling and Analysis using QPME: Queueing Petri Net Modeling Environment v2.0," in *Proc. of the 33rd Int. Conf. on Application and Theory of Petri Nets and Concurrency*. Springer-Verlag, 2012, pp. 388–397.

[4] P. G. Harrison and N. M. Patel, *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1993.

[5] R. Puigjaner, "Performance modelling of computer networks," in *Proc. of the 2003 IFIP/ACM Latin America conf. on Towards a Latin American agenda for network research*, ser. LANC '03. New York, NY, USA: ACM, 2003, pp. 106–123.

[6] S. Becker, H. Koziolek, and R. Reussner, "The Palladio component model for model-driven performance prediction," *Journal of Systems and Software*, vol. 82, no. 1, pp. 3–22, 2009.

[7] P. G. Harrison, C. M. Lladó, and R. Puigjaner, "A general performance model interchange format," in *Proc. of the 1st int. conf. on Performance evaluation methodolgies and tools*, ser. ValueTools '06. New York, NY, USA: ACM, 2006.

[8] A. Mitschele-Thiel and B. Müller-Clostermann, "Performance engineering of SDL/MSC systems," *Comput. Netw.*, vol. 31, no. 17, pp. 1801–1815, 1999.

[9] N. de Wet and P. Kritzinger, "Using UML models for the performance analysis of network systems," *Comput. Netw.*, vol. 49, no. 5, pp. 627–642, 2005.

[10] I. Dietrich, F. Dressler, V. Schmitt, and R. German, "SYNTONY: network protocol simulation based on standard-conform UML2 models," in *Proc. of the ValueTools '07*, 2007, pp. 21:1–21:11.

[11] "SDL combined with UML," ITU-T Z.109, International Telecommunication Union, 2000.

[12] A. Prakash, Z. Theisz, and R. Chaparadza, "Formal methods for modeling, refining and verifying autonomic components of computer networks," in *Transactions on Computational Science XV*. Springer, 2012, pp. 1–48.

[13] D. A. Zaitsev and T. R. Shmeleva, "A Parametric Colored Petri Net Model of a Switched Network," *Int. J. Communications, Network and System Sciences*, no. 4, pp. 65–76, 2011.

[14] G. Ciardo, L. Cherkasova, V. Kotov, and T. Rokicki, "Modeling a fibre channel switch with stochastic petri nets," *SIGMETRICS Perform. Eval. Rev.*, vol. 23, no. 1, pp. 319–320, 1995.

[15] L. Kristensen and K. Jensen, "Specification and validation of an edge router discovery protocol for mobile ad hoc networks," in *Integration of Software Specification Techniques for Applications in Engineering*, 2004, pp. 248–269.

[16] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann, "A Methodology for Performance Modeling of Distributed Event-Based Systems," in *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, 2008, pp. 13–22.

[17] S. Kounev, K. Bender, F. Brosig, N. Huber, and R. Okamoto, "Automated Simulation-Based Capacity Planning for Enterprise Data Fabrics," in *4th International ICST Conference on Simulation Tools and Techniques*, 2011, pp. 27–36.

[18] S. Kounev and C. Dutz, "QPME - A Performance Modeling Tool Based on Queueing Petri Nets," *ACM SIGMETRICS Performance Evaluation Review (PER), Special Issue on Tools for Computer Performance Modeling and Reliability Analysis*, vol. 36, no. 4, pp. 46–51, 2009.

[19] P. Rygielski, S. Zschaler, and S. Kounev, "A metamodel 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.

[20] P. Rygielski, S. Kounev, and S. Zschaler, "Model-Based Throughput Prediction in Data Center Networks," in *Proc. of the 2nd IEEE Int. Workshop on Measurements and Networking (M&N 2013)*, 2013, pp. 167–172.

[21] 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.

[22] F. Bause, "Queueing petri nets-a formalism for the combined qualitative and quantitative analysis of systems," in *Petri Nets and Performance Models, 1993. Proceedings., 5th International Workshop on*, 1993, pp. 14–23.

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

[24] 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.