A Meta-Model for Performance Modeling of Dynamic Virtualized Network Infrastructures*

[Work-in-Progress Paper]

Piotr Rygielski
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany
piotr.rygielski@kit.edu

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

Samuel Kounev
Karlsruhe Institute of Technology
Am Fasanengarten 5
Karlsruhe, Germany
kounev@kit.edu

ABSTRACT

In this work-in-progress paper, we present a new meta-model designed for the performance modeling of dynamic data center network infrastructures. Our approach models characteristic aspects of Cloud data centers which were not crucial in classical data centers. We present our meta-model and demonstrate its use for performance modeling and analysis through an example, including a transformation into OMNeT++ for performance simulation.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

Keywords

performance modeling, data center networks, meta-modeling

1. INTRODUCTION

The increasing popularity of Cloud Computing is leading to the emergence of large virtualized data centers hosting increasingly complex and dynamic IT systems and services. The performance of applications running in such data centers depends on the performance of three main subsystems: computing, storage, and networking. Different applications utilize these subsystems in different ways. The amount of computational, storage and communication resources assigned to applications influences significantly the quality-of-service (QoS) delivered to end users.

Due to the common adoption of virtualization technologies, Cloud 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 (SLAs) [2]. This dynamism of data center infrastructures turns an accurate and timely performance analysis into a real challenge [7].

In our research, we focus on network infrastructures of Cloud data centers. The main incentive for this is the fact that network infrastructures in virtualized and dynamic environments are introducing several new challenges for performance analysis. First, the growing density of Cloud data centers due to resource sharing leads to drastic increase in the number of network end points deployed on top of the same physical infrastructure. This situation makes performance analysis difficult due to, for example, high traffic volume from various sources traversing over a single physical link. Second, the volume of inner-data center traffic (having its source and destination inside the same data center) is much higher in Cloud environments than in classical data centers. Third, new traffic sources in the management layer of Cloud environments emerge (e.g., traffic caused by the migration of virtual machines). Finally, there is a lack of standardized and well established technology for network virtualization supporting performance isolation and QoS assurance. This requires performance models that are largely technology independent.

In this paper, we propose a new performance meta-model aiming for modeling virtualized network infrastructures addressing the above mentioned challenges. The meta-model is designed as a part of the Descartes Meta-Model (DMM) [10], a new meta-model for run-time QoS and resource management in virtualized service infrastructures. It covers, among others, computing infrastructures, storage, middleware, software and services. DMM is designed to serve as a basis for autonomic resource management during operation ensuring that system QoS requirements are continuously satisfied while infrastructure resources are utilized efficiently.

Some approaches for modeling and analyzing network performance in data centers already exist in the literature. We discuss a selection below. Most of the existing approaches (e.g., [3, 13]) are based either on black-box models or on highly-detailed protocol-level simulation models (e.g., [11, 4, 13]). The black-box models do not consider the internal network structure and topology while the protocol-level simulation models focus only on selected parts of the network infrastructure and do not capture the link to the running applications and services which are sources of the network traffic. Furthermore, existing modeling approaches do not

*This work is a part of RELATE project supported by the European Commission (Grant no. 264840ITN). We thank Jörg Henß for providing the meta-model of the OMNeT++.

ICPE'13, April 21–24, 2013, Prague, Czech Republic.

consider network virtualization techniques and their performance influences [14].

In [5], the authors proposed a modeling approach named Syntony. They use Syntony to model the Ad-hoc On-Demand Distance Vector protocol and compare the model-based analysis to the native OMNeT++ implementation. In [4], the authors use proSPEX (an approach based on the ITU-T Z.109 [1] profile) to evaluate the performance of the ESRO (Efficient Short Remote Operations) transport protocol. Both Syntony and proSPEX focus only on modeling specific network protocols. Approaches based on ITU-T SDL/MSC[1] (Specification and Description Language/Message Sequence Charts) have general focus and mainly model communication protocols [11]. The Palladio Component Model (PCM) [3] treats network infrastructures as a black-box and does not support any detailed network abstractions. GANA [12] (Generic Autonomic Network Architecture) models only autonomic components of networks on the protocol level. Additionally, in [13], the classical black-box and low-level simulation based performance modeling approaches in the area of computer networks are described.

We stress that none of these works explicitly take into consideration the influences of network virtualization techniques on the system performance and the mentioned specific aspects of Cloud data center networks.

2. ENVISIONED APPROACH

We propose a model-based approach to performance analysis. The networking domain of data centers is described by a new meta-model presented in this paper. The DNI Meta-Model (*Descartes Network Infrastructure*) is the base for DNI models instantiation. A DNI model can utilize external information about the deployed software and the traffic it produces. The general approach is depicted in Figure 1.

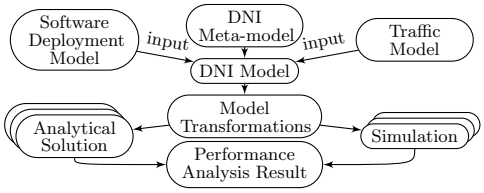

Figure 1: The approach diagram with model inputs.

The obtained DNI model is transformed into known performance models (e.g., queueing networks, Petri nets) and analyzed using existing performance analysis methods. As the research develops, DNI models can be transformed into multiple performance models and deliver multiple performance analysis results, e.g., with different accuracy.

As our work on the meta-model is still in progress, the current version of the DNI Meta-Model does not cover all important, performance-influencing aspects of data center network infrastructures. Our development roadmap includes expansion of the meta-model by adding entities covering, e.g., QoS assurance, routing, network virtualization by tunneling and overlaying. In the next section, we present the DNI Meta-Model in its current state of development.

3. META-MODEL

The DNI Meta-model[1] covers four parts of data center network infrastructures: network protocols, structure of the physical and virtual network infrastructure (nodes, interfaces, links), deployed virtual machines and applications (traffic sources), and a part that describes the network traffic in the data center. The meta-model is implemented in Ecore using the Eclipse Modeling Framework (EMF).

Network protocols control the process of any data exchange between communicating nodes. They define means of identification of communicating parties; that is, addresses of the source and destination. Additionally, the implementation details of a given protocol have strong influence on the performance of data transmission. In the DNI Meta-Model, the `NetworkProtocols` are organized in `NetworkProtocolStacks` that allows the application of a complete stack to a given part of the network. Each protocol can provide a `ProtocolAddress`—for example, to a `NetworkInterface`. This part of the DNI Meta-Model is presented on the class diagram depicted in Figure 2.

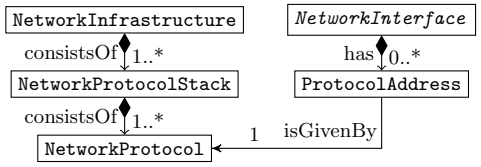

Figure 2: Network protocols and protocol stacks.

The protocols are described in the meta-model with the information about introduced overheads. Due to the wide variety of protocols used nowadays, we have captured only common, generic features describing a protocol, e.g., data unit header size, and whether communication is connection-oriented.

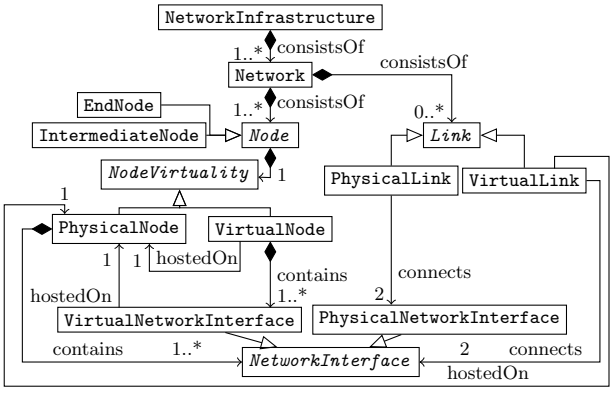

Figure 3: Network's physical and virtual structure.

The part of the meta-model representing the physical and virtual structure of a network is depicted in Figure 3. The `NetworkInfrastructure`, apart from the `NetworkProtocolStack`, consists of `Networks`. Each `Network` is built from `Links` and `Nodes` which are connected by `NetworkInterfaces`. As their impact on network performance differs, `Nodes` are categorized as `EndNode` (e.g., server, VM, gateway to an external network) or `IntermediateNode` (e.g., switch, router, firewall). Each of the `Network` components can be further characterized as physical or virtual where each virtual entity must be hosted on a physical one.

The information about physical to virtual resource allocations of `EndNodes` is included in the meta-model in a simpli-

[1]Available online: `http://bit.ly/DNI-model-ICPE2013`

fied way—a virtual entity gets a percentage of the physical resources (removed from Fig. 3 for clarity)—although the data is assumed to be imported from other parts of DMM.

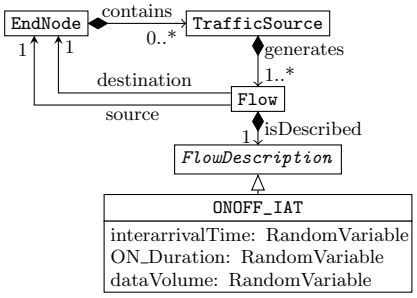

Figure 4: Network traffic and software deployment.

In a data center, most of the network traffic is generated by deployed applications. As depicted in Figure 4, network traffic is generated by `TrafficSource`s that are deployed on `EndNode`s. Each `TrafficSource` generates `Flow`s which have exactly one source and destination located in the `EndNodes` and can be uniquely identified by the set of protocol-level addresses. For now, we assume that there is exactly one route from the source to destination. Finally, each `Flow` can be described using various models. In this paper, we provide the classical description of an ON-OFF source with inter-arrival times specified by a probability distribution. However, a variety of other traffic models can be found in the literature (e.g., [8]) which can be integrated into our meta-model.

4. EXAMPLE

In this section, we first present a selected fragment of an exemplary data center network infrastructure which is modeled with DNI. Next, to obtain performance analysis results, we apply a model-to-model transformation of the DNI instance into an instance of OMNeT++ simulation [6, 15]. In this example, selected parts of the transformation have manually hard-coded performance impacts, however there exist systematic relations between the source and destination model and the hard-coded values will be removed in a future version of the transformation.

In the considered scenario (cf. Fig. 5a), the network consists of three servers and one switch. On two servers two virtual machines (VMs) are deployed. Each of the VMs is connected internally to a software hypervisor switch. The servers are connected to the physical switch using an IP network. Fig. 5b shows part of the model of this scenario (focusing on only one server). The complete model, the meta-model, the transformation and its result can be found online (see the URL in Section 3). The network infrastructure depicted in Figure 5 lacks the information about any software deployed in the machines as this information can be provided manually (e.g., from measurements) or imported from other state-of-the-art models (e.g., PCM [3] or DMM [10]) using model transformations.

The entities from the DNI model are transformed into entities recognized by OMNeT++ using the transformation rules described in Table 1. In the destination model, we use the `SimpleHypervisor`[2] module that is a modification of the `StandardHost` in order to enable sharing of a physical network interface among all hosted VMs. The transformation has been implemented using the Epsilon Transformation Language [9]. Below we discuss two excerpts from the transformation code in more detail.

In Listing 1, we show a fragment of the transformation code describing how the generic features of network protocols are translated into overheads in data transmission. We assume that overheads of protocols of lower layers (i.e., these addressing a *`NetworkInterface`*) are decreasing the capacity of a link transmitting data. The overhead of higher layer protocols (e.g., TCP that addresses an application) increase the amount of data produced by a `TrafficSource`. In the future, we plan to abstract from the classical ISO/OSI layers and support layering in a generic fashion.

Listing 1: Utilizing network protocols information in the model transformation (language: ETL).

```
operation CreateChannels
var p=stack.getProtocols();
var headLen=p.select(pr|pr.layer=2).headerLenght;
headLen=headLen+p.select(pr|pr.layer=3).headerLenght;
var l2UnitSize=p.select(pr|pr.layer=2).dataUnitSize;
var overhead = (100*headLen)/l2UnitSize;
var channel = new OMNET!Channel;
var par = new OMNET!DoubleParameter;
par.name = "datarate";
var realThr=link.maxThroughput*(1.0-(overhead/100.0));
par.value = realThr.equivalent();
par.unit = link.unit.equivalent();
channel.parameters.add(par);
```

In Listing 2, we demonstrate how the physical resources are shared among virtual `EndNode`s bound by *hostedOn* relation. Using the simplified resource allocation information, the total bandwidth available to a physical node is divided among virtual nodes using the percentage value and a new transmission delay is calculated.

Listing 2: Sharing physical resources among virtual entities.

```
operation networkAddConnections
for(link in DNI!VirtLink.allInstances){
 var physRes=link.hostedOn.getTotalBandwidth();
 var conn=new OMNET!ChannelConnectionItem;
 var ch=OMNET!Channel.allInstances.selectOne(
    c|c.name==link.name);
 var delay=ch.parameters.selectOne(p|p.name=="delay");
 var rate=ch.parameters.selectOne(p|p.name=="datarate");
 var l2=DNI!NetworkProtocol.allInstances.selectOne(
    p|p.layer=2);
 var resAlloc=link.hostedOn.hasResAlloc;
 var ra=resAlloc.selectOne(r|r.virtNode==conn.leftNode);
 var nDelay=l2.dataUnitSize/(rate.value*ra.resShare);
 ch.setDelay(delay.value + nDelay);
 ch.setDatarate(0.0);
 conn.channel=ch; }
```

5. CONCLUSIONS

In this work-in-progress paper, we have presented a new meta-model aiming for performance modeling of virtualized data center network infrastructures. Our meta-model addresses the challenges of performance analysis in Cloud environments by abstracting too detailed and technology-specific information, while covering important performance-relevant aspects of network infrastructures. We have presented an example of model transformation to a popular simulation framework where performance analysis can be run. Our future research steps concern primarily further development of the meta-model to support resource allocation, QoS aspects, and network virtualization techniques. Furthermore,

[2]Available online; See the URL in Section 3.

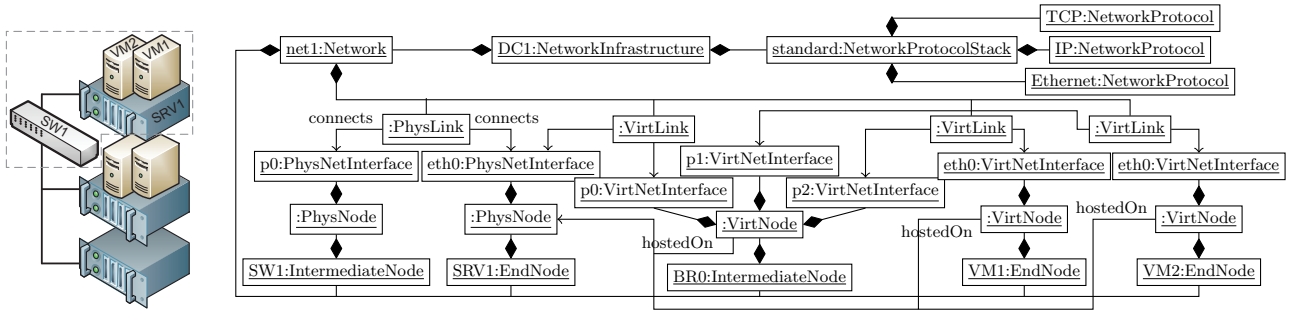

Figure 5: Selected fragment of an exemplary data center network (a) presented on the object diagram (b).

Table 1: Selected transformation rules from DNI to OMNeT++

Source Entity	Destination Entity	Comments
`NetworkProtocolStack`	a parameter of a `DatarateChannel`	Protocols used in the stack introduce transmission overheads. An excerpt of the transformation code is presented in Listing 1.
`NetworkInfrastructure`	`network`	A `network` in OMNeT++ represents the whole network infrastructure. DNI `Network`s are distinguished by protocol-level addresses.
`EndNode`	`SimpleHypervisor`	Applies to both `PhysicalNode` and `VirtualNode`.
`IntermediateNode`	e.g., `Switch` or `Router`	Depends on the value of `IntermadiateNode.type` enumeration.
`Link`	`DatarateChannel`	Applies to both `PhysicalLink` and `VirtualLink`.
`NetworkInterface`	`inout` gate in e.g., `Router`.	Applies to the `gates` section of a *`Node`* the gate is attached to. Applies to both `Physical-` and `VirtualNetworkInterface`.
`TrafficSource`	e.g., `TcpApp`, `UdpApp`	Choice of application type depends on the `NetworkProtocol` in the L4 (e.g., for TCP we use `TcpApp`).
`FlowDescription`, `ONOFF_IAT`	configuration of e.g., `TcpApp`	The choice of concrete *`FlowDescription`* depends on traffic generation implementation and configuration of `TrafficSources`.
mapping of `Virtual-` on `Physical-` entities	a parameter of a `DatarateChannel`	Sharing of physical resources causes modification of virtual links throughput. See transformation code in Listing 2.

we plan to provide more ways of traffic modeling and integrate the approach with DMM.

6. REFERENCES

[1] SDL combined with UML. ITU-T Z.109, 2000.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical report, University of California, 2009.

[3] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3–22, 2009.

[4] N. de Wet and P. Kritzinger. Using UML models for the performance analysis of network systems. *Comput. Netw.*, 49(5):627–642, 2005.

[5] 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*, pages 21:1–21:11, 2007.

[6] J. Henss. OMPCM – An OMNeT++ simulator for Palladio. In *Palladio Days 2012*, Paderborn, Germany, 2012. Special Focus Talk.

[7] 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*, pages 563–573, 2011.

[8] T. Karagiannis, M. Molle, and M. Faloutsos. Long-range dependence: Ten years of internet traffic modeling. *IEEE Internet Computing*, 8(5):57–64, 2004.

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

[10] S. Kounev, F. Brosig, and N. Huber. Descartes Meta-Model (DMM). Technical report, Karlsruhe Institute of Technology, 2013.

[11] A. Mitschele-Thiel and B. Müller-Clostermann. Performance engineering of SDL/MSC systems. *Comput. Netw.*, 31(17):1801–1815, 1999.

[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*, pages 1–48. Springer, 2012.

[13] 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*, LANC '03, pages 106–123, New York, NY, USA, 2003. ACM.

[14] P. Rygielski and S. Kounev. Network Virtualization for QoS-Aware Resource Management in Cloud Data Centers: A Survey. *PIK — Praxis der Informationsverarbeitung und Kommunikation*, 36(1), 2013. In print.

[15] A. Varga. The OMNeT++ discrete event simulation system. In *Proc. of the European Simulation Multi-conference (ESM)*, pages 319–324, 2001.