Towards Virtualization of Software-Defined Networks: A Journey in Three Acts

Andreas Blenk
Technical University of Munich
Munich, Germany
andreas.blenk@tum.de

Wolfgang Kellerer (supervisor)
Technical University of Munich
Munich, Germany
wolfgang.kellerer@tum.de

***Abstract*—Today's networks lack the support to satisfy the highly diverse and fast changing demands of emerging applications and services. The paradigms Network Virtualization (NV) and Software-Defined Networking (SDN) can potentially overcome this impasse. The virtualization of software-defined networks is expected to bring dynamic resource sharing with guaranteed performance through NV and programmability through SDN; for the first time, tenants can program their requested network resources according to their service demands in a timely manner. However, the virtualization of SDN-based networks introduces new challenges for operators, e.g., a virtualization layer that provides low and guaranteed control plane latencies for tenants. Moreover, tenants' expectations range from a fast, nearly-instantaneous provisioning of virtual networks to predictable operations of virtual networks. With this paper, we give a comprehensive overview of the thesis, which can be split into three parts — a journey in three acts. The thesis first presents a measurement procedure and a flexible virtualization layer design for the virtualization of software-defined networks. Focusing on the control plane, it introduces mathematical models for analyzing four virtualization layer architectures. Third, for a fast and efficient virtual network provisioning on the data plane, the thesis proposes optimization systems using Machine Learning and Neural Computation.**

***Index Terms*—Network Virtualization, Software-Defined Networking, Network Measurements, Optimization, Machine Learning, Neural Computation**

I. Introduction

Existing problems: Network traffic of today's applications, like in cloud-computing or machine-to-machine communication networks, can fluctuate on time scales from minutes to milliseconds and are possibly highly diverse with respect to their required network resources [28]–[31]. However, communication networks with their current protocol stacks lack adequate mechanisms to handle changing application requirements in a timely manner. Hence, today's communication networks lack the flexibility in providing efficient resource sharing with a high level of adaptability, needed to support demands with diverse network resource requirements changing over time. Overall, this can result in a performance that is far from perfect for the network operator and the network users.

Two paradigms, namely Network Virtualization (NV) [32] and Software-Defined Networking (SDN) [33], are expected to cope with requirements for flexible network resource sharing and adaptability. Whereas NV is seen as a key enabler to overcome the ossification of the Internet by introducing flexible resource sharing [32], SDN introduces a new way of flexibly programming the shared resources at runtime [33]. NV abstracts physical resources of Infrastructure Providers (InPs) and enables tenants, i.e., Service Providers (SPs), to use virtual resources according to their users' demands. Due to NV, both InPs and SPs are expected to control physical and virtual resources in a dynamic and independent manner. SDN decouples the control plane (CP) of network devices, such as routers and switches, from their data plane (DP). Using open interfaces and protocols such as OpenFlow (OF), SDN provides new means of network programmability [33]. With networks being completely programmable, SDN can realize control logics, i.e., network operating systems (NOSs) integrating new concepts; new NOSs can be tailored to application-, service-, and user-specific demands.

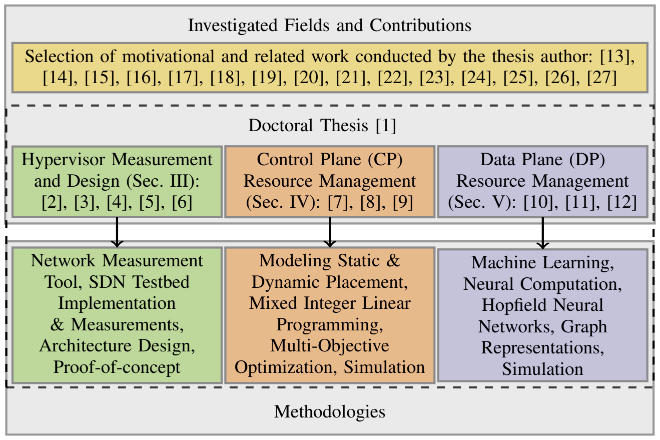
Investigated Fields and Contributions
Selection of motivational and related work conducted by the thesis author: [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]
Doctoral Thesis [1]
Hypervisor Measurement and Design (Sec. III): [2], [3], [4], [5], [6]
Control Plane (CP) Resource Management (Sec. IV): [7], [8], [9]
Data Plane (DP) Resource Management (Sec. V): [10], [11], [12]
Network Measurement Tool, SDN Testbed Implementation & Measurements, Architecture Design, Proof-of-concept
Modeling Static & Dynamic Placement, Mixed Integer Linear Programming, Multi-Objective Optimization, Simulation
Machine Learning, Neural Computation, Hopfield Neural Networks, Graph Representations, Simulation
Methodologies

Fig. 1. Thesis structure - motivational and related work conducted by the author and the main investigated and contributed fields of the thesis. Thesis publications are classified into three chapters: hypervisor measurement and design, Control Plane (CP) resource management, and Data Plane (DP) resource management. Different methodologies are applied in each area. Whereas the first content chapter focuses on practical methods, both resource management chapters focus on simulation-based studies of optimization problems related to combining SDN and NV.

Potential solution: Combining NV and SDN is expected to offer the advantages of both worlds: a flexible and dynamic resource acquisition by tenants through NV and a standardized way to program physical and virtual resources through SDN. This is called *the virtualization of software-defined networks*, leading to multiple virtual software-defined networks (vSDNs) sharing one infrastructure [34], [35]. In such setting, multiple vSDNs coexist while each one is individually managed by its own network operating system through SDN. The combination should make it possible to implement, test, and even introduce new control logic at runtime into existing networking infrastructures. However, to realize the combination of SDN and NV, a so-called virtualization layer becomes necessary [36]. This virtualization layer is realized by one or many *SDN network hypervisors* or briefly also *hypervisors* in this paper. Interestingly, the virtualization layer itself can lead to unpredictable network management and control. For instance, as hypervisors are processing control plane packets of tenants, they can add additional processing latency even with high variation in case of high-load scenarios. Hence, tenants could perceive less predictable network CP latency.

Contribution of this paper: This paper provides an overview of the thesis [1], which focuses on addressing problems and challenges arising when combining NV and SDN. *The main research questions* addressed in the thesis, for which we will comprehensively survey potential answers in Sections III—V, are:

- How to measure and design a virtualization layer for software-defined networks? (Sec. III)
- How to make a virtualization layer more flexible and still predictable? (Sec. III)
- How to distribute a virtualization layer and identify its performance overhead? (Sec. IV)
- How to plan a distributed virtualization layer in case of dynamics? (Sec. IV)
- How to speed up the provisioning of virtual networks? (Sec. V)
- How to improve the efficiency of algorithms for virtual network provisioning? (Sec. V)

Fig. 1 illustrates the structure of the thesis [1] with respect to research areas and methodologies. The figure first shows the contributions outlined in the thesis in other research areas, e.g., application-aware networking or softwarization of mobile core networks, that serve as motivational work for the thesis [13]–[27]. Due to page limitations, we omit a deeper presentation of the motivational work. To address the aforementioned research questions, the surveyed thesis is split into three research areas: (1) measurements and design of SDN network hypervisor architectures, (2) mathematical models for the static and dynamic placement of hypervisors, (3) systems improving the efficiency of algorithms, e.g., Virtual Network Embedding (VNE) or SDN controller placement algorithms, needed for provisioning virtual networks. This paper summarizes the conducted research contributions of the thesis [1] (research areas (1) – (3)). For each part, it provides brief examples and explanations. For more details on the results, we refer the reader to the original thesis [1] and the published papers.

Structure of this paper: Sec. II outlines the challenges arising when combining NV and SDN. Afterwards, the paper focuses on the selected contributions outlined in the thesis structure within three sections: Sec. III outlines measurements and design for virtualizing software-defined networks, Sec. IV analysis optimization aspects of a virtualization layer with respect to the placement of hypervisor entities, and Sec. V outlines optimization systems for data plane resource provisioning algorithms relying on Machine Learning (ML) and Neural Computation. Sec. VI concludes the paper with a discussion on how the thesis addressed the aforementioned research questions.

II. Challenges when Combining NV and SDN

Within this section, we outline the challenges arising when combining NV and SDN with respect to the three research areas (see Fig. 1).

Like in computer virtualization where a hypervisor manages virtual machines and their physical resource access [37], a so-called virtualization layer consisting of one or many distributed hypervisors realizes the virtualization of SDNs [34], [36]. A hypervisors acts as a proxy between the controllers of tenants (implementing the tenants' control logic) and the shared physical infrastructure, where the vSDNs reside. The main hypervisor functions are translation, abstraction, and CP and DP isolation. The tenant controllers establish CP connections with their virtual switches via the hypervisor and its functions. The translation function translates the CP messages between tenant controllers and the virtual SDN networks; the abstraction function provides an abstracted topology view of the physical SDN network; CP and DP isolation functions guarantee isolated and predictable performance on both CP and DP of the virtualized infrastructure. Interestingly, a hypervisor itself can introduce overhead (e.g., CP latency) and performance interference among tenants. In contrast to SDN without virtualization, tenants can interfere on the shared control plane resources of the virtualization layer implementation, i.e., of the hypervisor functions (translation, abstraction, and isolation). Hence, a deep understanding of design choices and performance implications of hypervisor implementations is of significant importance. Such analysis, however, demands for emulating potentially multiple tenant controllers and switches simultaneously in contrast to non-virtualized networks.

Beside implementation aspects of vSDNs, resource planning and management is another challenging task when combining NV and SDN. With SDN, the control logic (i.e., the SDN controllers) can be flexibly distributed among the network [38]. Similar, network hypervisors implementing the logic for virtualization can also be distributed among the network. Since the tenant CP traffic needs to pass through hypervisors, the placement of network hypervisors is crucial for providing low and predictable CP latencies for tenants. Placing network hypervisors, however, is an algorithmically hard problem. In contrast to the placement of SDN controllers, it even adds a new dimension: hypervisors must serve multiple vSDNs. Moreover, in case of dynamic scenarios where virtual networks arrive over time, the virtualization layer, i.e., the hypervisor placements, might need to change for achieving an always low CP latency. Adapting the virtualization layer can introduce reconfigurations that should be considered; neglecting reconfigurations can lead to network instabilities resulting in service interruptions or network outages [39]–[42].

The virtualization of software-defined networks also introduces new challenges with respect to resource allocation on the DP: vSDNs can be requested and instantiated by SPs at any time. The allocation of arriving requests, i.e., the embedding of virtual networks, needs to be solved fast and efficiently for an overall optimal network performance: e.g., fast and low-cost acquisition is a competitive advantage when virtualizing an infrastructure like in cloud environments [43], [44]. However, solving the VNE problem is algorithmically hard as well [45]. Hence, managing virtualization layers requires new mechanisms to improve the embedding quality of virtual networks and to speed up existing embedding algorithms.

In the following three Sections III — V, we comprehensively summarize how the thesis [1] addresses the aforementioned challenges with respect to each area.

III. Act I: Measurements and Design for Virtual Software-Defined Networks

This section summarizes the first contribution chapter of the thesis [1]. The chapter starts by analyzing existing SDN network hypervisor architectures and outlines their shortcomings. As there has been no measurement framework for SDN hypervisors in literature so far, the chapter presents a comprehensive benchmarking framework for making measurement-based analysis of hypervisors. To conduct hypervisor measurements, the tool *perfbench* is presented [6], [16]. In order to mitigate the sources of unpredictability in existing hypervisor architectures, this chapter outlines and implements concepts for isolation and CP adaptation support [2]–[4].

A. Comprehensive Survey of SDN Network Hypervisors

The thesis [1] conducts a survey of existing hypervisors with respect to their implementations of hypervisor functions (translation, abstraction, isolation) [5]. Furthermore, the network hypervisors are classified into centralized and distributed hypervisor architectures. The main centralized hypervisor is FlowVisor [34], which built the basis for many upcoming hypervisor proposals. Distributed hypervisor architectures are further subclassified according to their execution platforms: hypervisors that only rely on general computing platforms, hypervisors that additionally make use of the features of general purpose networking elements, and hypervisors that additionally use network elements that are adapted to the virtualization of SDN networks. Generally, hypervisor architectures whose virtualization functions can be distributed are seen as most flexible. For instance, isolation functions can be implemented on network elements while translation functions can remain on software implementations hosted on virtual machines. Thereby, isolation functions can protect the performance of translation functions, e.g., by delaying CP messages of tenants. Although there exists a wide range of hypervisor designs, we observed the following shortcomings from our analysis: (1) the lack of a comprehensive benchmarking framework to quantify performance and (2) the lack of a virtualization layer design that provides flexible virtualization mechanisms with guaranteed network performance.

B. Measurement Tool and Study of SDN Network Hypervisors

In order to overcome the lack of a comprehensive performance benchmark tool for hypervisors, the thesis propose *perfbench* for OF-based SDN network hypervisors [6], [46]. The tool satisfies the requirements for hypervisor benchmarks, e.g., for datacenters [28]: controllable-high, stable, and variable OF message rates as well as the emulation of multiple tenants and switches simultaneously. Moreover, *perfbench* can be used in non-virtualized SDN networks for switch performance investigations [16]. With *perfbench* (publicly available [47]), we find that hypervisors differently impact on the vSDN performance under varying workloads. For instance, FlowVisor [34] adds less latency overhead than OpenVirteX [48]. For OF FlowMod messages with a rate of $10\,000$ and $30\,000$ messages per seconds, FlowVisor adds latency overheads from $0.4\,\mathrm{ms}$ to $1\,\mathrm{ms}$, whereas OpenVirteX adds $1\,\mathrm{ms} - 30\,\mathrm{ms}$. However, given the same processing resources, OpenVirteX can support larger networking scenarios with up to 100 tenant controllers and switches, whereas FlowVisor can only support up to 20 controllers and switches simultaneously.

C. Network Virtualization Architecture Towards Predictable CP Performance: `HyperFlex`

The hypervisor measurements reveal CP interference among tenants [1], [6]. As a consequence, the thesis proposes HyperFlex - a flexible virtualization layer architecture with improved predictable CP performance for tenants [2]–[4]. HyperFlex (publicly available [49]) decomposes the virtualization layer into software and hardware-based functions; the functions (i.e., abstraction, translation, and isolation) can be flexibly distributed among servers or networking hardware. For more predictable and flexible network management, two mechanisms are implemented: an adaptation mechanism [3] and a CP isolation concept [2], [4].

1) CP Isolation: In vSDNs, tenants are sharing the CP resources of hypervisors and switches. Hence, the CP messages sent by multiple tenant controllers can interfere on the shared hypervisor resources. For instance, the messages of one tenant can overload the translation function, which leads to potentially high CP latency for all other tenants using the same translation function. Using the isolation functions, CP latency guarantees can be achieved by dropping or delaying CP messages, e.g., of malicious tenant controllers. HyperFlex provides two isolation function implementations: a software-based implementation and a hardware-based implementation that uses traffic policing mechanisms of networking hardware. The measurement studies on the CP isolation reveal trade-offs between software and hardware implementations: their usage depends on the OF message type and current infrastructure utilization. For instance, hardware isolation on network elements leads to network packet drops. In contrast, dropping OF messages in software can protect either hypervisor or switch performance. However, for proper use, software isolation demands a headroom reservation of CPU resources since it is processed by the hypervisor CPU.

2) CP Reconfiguration: The adaptation mechanism of HyperFlex addresses the handover of control plane connections between hypervisor functions and SDN switches. This mechanism can help, e.g., to overcome overload situations by offloading CP connections from one virtualization function to another. Measurements of the adaptation mechanism show that the CP assignments of switches to hypervisor functions can be changed at runtime [15]. This comes, however, at the cost of increased CP latency during the migration process: for instance, the CP latency can increase from $30\,\mathrm{ms}$ to $40\,\mathrm{ms}$ in an exemplary networking scenario with $1\,000$ CP messages per second. As adaptations induce additional latency, the dynamic hypervisor placement model will also focus on minimizing reconfigurations (see Sec. IV).

IV. Act II: Modeling and Optimization of Network Virtualization Layers

This section briefly summarizes the second contribution chapter of the thesis [1]. Since no analysis of the impact of the hypervisor placement exists yet, the thesis chapter proposes mathematical models for optimizing network hypervisor placements for static and dynamic use [7]–[9]. The mathematical models can help answer questions such as how many hypervisors are needed and where to distribute them among network locations.

A. Hypervisor Placement for Static Use

The thesis chapter starts with the investigation of the placement of hypervisor instances for static use — the set of vSDNs will not change over time [7], [8]. We define mixed-integer linear programming (MILP)-based models for a centralized and three distributed hypervisor architectures. The distributed hypervisor architectures can either not use, partially use, or fully use special hardware features of the infrastructure. In particular, we investigate the impact of multi-controller switches that can simultaneously connect to multiple hypervisor instances. For evaluation of the four architectures, we investigate the impact of the hypervisor placement on the CP latencies of the entire network as well as individual vSDNs. In order to account for the individual vSDNs, two virtualization-related CP latency measures are defined: maximum-average and average-maximum. The measures are defined to particular count for the virtualization use case: the maximum-average measure addresses the maximum of all average CP latencie of vSDNs whereas the average-maximum addresses the average of all maximum CP latencies of vSDNs. This extends the basic CP latencies, namely average and maximum CP latency, of controller placements in non-virtualized networks [38]. We identify the CP latency overhead due to the requirement for the tenants' control connections to traverse a hypervisor instance: we call this *the cost of virtualization* [8].

Exemplary observations: virtualization can add significant CP latency overhead for individual vSDNs. For instance, with only one hypervisor, the (average/maximum) CP latencies for $75\ \%$ of vSDNs can increase by a factor of 2 compared to an non-virtualized SDN network. We also show that for wide area networks with 30 to 50 nodes, seven hypervisor instances and $50\,\%$ switches supporting the multi-controller feature leads to no overhead for more than $75\ \%$ of vSDNs, i.e., CP latencies are equal to the non-virtualized case.

B. Hypervisor Placement for Dynamic Use

We extend the study of the hypervisor placement for a static use case to dynamic use: a virtual network request arrives over time and needs to be served by the virtualization layer [9]. We extend the MILP model for static use to count for reconfigurations. We model the update of CP connections and the migration of network hypervisors as reconfigurations. As minimizing reconfigurations and minimizing latency are conflicting objectives, the ϵ-constraint method is used to solve the multi-objective optimization problem [50]. We analyze the trade-off between reconfigurations and latency and the impact of relaxing the latency for reducing reconfigurations. Relaxing latency represents *the cost of adaptation*; a new angle when looking at dynamic hypervisor deployments.

We first notice that average CP latencies change with a probability of $75\ \%$ when a new virtual network arrives, whereas maximum control latencies change only with a probability of roughly $25\ \%$ when a new virtual network arrives. The observation might lead to a false hypothesis: maximum objectives might not demand reconfigurations. However, for three hypervisors, up to $40\ \%$ of vSDNs are affected by reconfigurations for maximum and average objectives. Relaxing the latency can help to reduce the amount of hypervisor location changes; however, for maximum objectives, it is harder to avoid reconfigurations.

V. Act III: Machine Learning-based Algorithm Preprocessing for Virtual Network Provisioning

The third content chapter of the thesis [1] investigates the potential of Machine Learning (ML) and Neural Computation to provide more efficient and faster virtual network provisioning. It first presents *NeuroViNE*, a system that uses Neural Computation [11]. Second, it outlines *o'zapft is*, an ML approach that leverages the available data of previously solved problem instances to improve the efficiency of future executions of network algorithms [10], [12].

A. Preprocessing Using Neural Computation: `NeuroViNE`

NeuroViNE is a preprocessor for VNE algorithms [11]. *NeuroViNE* is motivated by the observation that efficient solutions to the VNE problem place frequently communicating nodes close to each other. Accordingly, *NeuroViNE* extracts subgraphs that provide (1) a high probability for being able to accommodate a virtual network and (2) ensure a low-cost embedding. In order to find valuable subgraphs, *NeuroViNE* uses a Hopfield neural network [51], [52].

Based on our simulation results, we observe that existing time-intensive VNE algorithms benefit if they can be executed on subgraphs selected intelligently from the substrate network (wide area networks, datacenter, and random networks): *NeuroViNE* can potentially shorten network algorithm runtimes while preserving a high embedding quality, e.g., by more than 80 % for virtual networks with 15 nodes on a 100 node substrate. Moreover, *NeuroViNE* reduces the embedding cost of the GRC algorithm [53] by roughly 25 %.

B. Data-Driven Algorithm Optimization: `o'zapft is`

This section presents *o'zapft is*, an ML-based system that shows how to exploit the wealth of data generated by algorithms [12]. *o'zapft is* relies on supervised learning. In the thesis [1], *o'zapft is* is applied to the VNE problem and the SDN controller placement problem. Based on the collected data from previous embedding solutions (i.e., whether requests can be embedded or their embedding quality), *o'zapft is* learns to predict the outcome of embedding algorithms. For the embedding of networks, using this knowledge, hard-to-model problem instances can be rejected without triggering a runtime expensive embedding algorithm. For the placement of tenant SDN controllers, *o'zapft is* learns from previous placement solutions to provide initial solution guesses for future placement problems. By predicting the chance of a node to host a controller, *o'zapft is* preselects valuable network nodes to create initial solutions. Thereby, it can shrink the search space before algorithms effectively search for controller locations.

By making prediction on the feasibility of arriving virtual networks, the *o'zapft is* framework reduces the runtime of an exact VNE algorithm [54] by roughly 50 % on a 100 node substrate network. When placing SDN controllers for tenants, *o'zapft is* preselects 50 % of substrate nodes as candidate nodes. When executing an exact algorithm on the subset of candidate nodes, still the best possible latency objective can be achieved while effectively shrinking the search space; hence, *o'zapft is* saves computational algorithm cost and speeds up the execution time.

VI. Discussion and Outlook

Virtualizing SDN networks introduces new problems and challenges. Indeed, our SDN network hypervisor measurements and analysis reveal that the virtualization itself introduces unpredictability. With our tool *perfbench*, hypervisors can now be quantified in much more detail; hence, more precise performance models can be created and sources of unpredictability can be detected. We believe that such knowledge is important when dimensioning and designing virtualization layers for SDN networks. As a step towards more predictable virtualization layer management, *HyperFlex* can better isolate the control plane performance of tenant controllers. Furthermore, with its adaptation mechanism, it can also prevent SDN network hypervisors from over-utilization. For future, we believe that more focus should be put on potential interference on CP and DP in virtualized SDN networks.

When looking at the distribution of SDN network hypervisors, we introduce optimization models that let network operators rigorously examine the trade-offs between using different numbers of SDN hypervisor instances and multi-controller SDN switches. The static models are additionally extended to more dynamic use cases where virtual networks arrive over time. Revealed as sources of performance overheads (i.e., additional control plane latency), reconfigurations could be efficiently reduced. However, this comes at the cost of additional control plane latency during operation. Again, operators should take this trade-off into account when dimensioning virtualization layers for dynamic scenarios. As a next step, new models should also analyze the capacitated case of the investigated problems.

When virtualizing SDN networks, the algorithmically hard VNE problem needs to be addressed. To fully unleash the advantages of both NV and SDN, new (heuristic) ways of improving the algorithm performance and embedding quality of virtual networks are needed. Here, our simulation-based results show that Machine Learning and Neural Computation offer new optimization opportunities. The runtime and solution quality of existing VNE algorithms is rigorously improved. Using our solutions, exact algorithms based on mathematical programming can become credible alternatives to heuristics on larger topologies. Moreover, as more general problems are underlying the embedding problem, our concepts could be extended to other research areas (network protocol design, scheduling etc.).

Acknowledgment

This work has been performed in part in the framework of the CELTIC EUREKA project SENDATE-PLANETS (Project ID C2015/3-1) funded by the German BMBF (Project ID 16KIS0473), and in part in the framework of the EU project FlexNets funded by the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets).

References

[1] A. Blenk, "Towards virtualization of software-defined networks: Analysis, modeling, and optimization," Dissertation, Technische Universität München, München, 2018. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20180302-1402009-1-5

[2] A. Basta, A. Blenk, Y.-T. Lai, and W. Kellerer, "HyperFlex: Demonstrating control-plane isolation for virtual software-defined networks," in *Proc. IFIP/IEEE Int. Symp. IM Netw.*, May 2015, pp. 1163–1164.

[3] A. Basta, A. Blenk, H. Belhaj Hassine, and W. Kellerer, "Towards a dynamic SDN virtualization layer: Control path migration protocol," in *Proc. IFIP/IEEE CNSM*, Nov. 2015, pp. 354–359.

[4] A. Blenk, A. Basta, and W. Kellerer, "HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation," in *Proc. IFIP/IEEE Int. Symp. IM Netw.*, May 2015, pp. 397–405.

[5] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on Network Virtualization Hypervisors for Software Defined Networking," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685, Firstquarter 2016.

[6] A. Blenk, A. Basta, L. Henkel, J. Zerwas, W. Kellerer, and S. Schmid, "perfbench: A tool for predictability analysis in multi-tenant software-defined networks," in *Proc. ACM SIGCOMM 2018 Conference on Posters and Demos*, New York, NY, USA, 2018, pp. 66–68.

[7] A. Blenk, A. Basta, J. Zerwas, and W. Kellerer, "Pairing SDN with network virtualization: The network hypervisor placement problem," in *Proc. IEEE NFV-SDN*, Nov. 2015, pp. 198–204.

[8] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control Plane Latency With SDN Network Hypervisors: The Cost of Virtualization," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 13, no. 3, pp. 366–380, Sep. 2016.

[9] J. Zerwas, A. Blenk, and W. Kellerer, "Optimization Models for Flexible and Adaptive SDN Network Virtualization Layers," *Computer Research Repository (CoRR)*, no. V, p. 1–4, Nov. 2016.

[10] A. Blenk, P. Kalmbach, P. V. D. Smagt, and W. Kellerer, "Boost Online Virtual Network Embedding : Using Neural Networks for Admission Control," in *Proc. IFIP/IEEE CNSM*, Oct. 2016, pp. 10–18.

[11] A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, and W. Kellerer, "NeuroViNE: A Neural Preprocessor for Your Virtual Network Embedding Algorithm," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 405–413.

[12] A. Blenk, P. Kalmbach, S. Schmid, and W. Kellerer, "*o'zapft is:* tap your network algorithm's big data!" in *Proc. ACM SIGCOMM Workshop Big-DAMA*, Aug. 2017, pp. 19–24.

[13] A. Blenk and W. Kellerer, "Traffic pattern based virtual network embedding," in *Proc. ACM CoNEXT Student Workshop*, Dec. 2013, p. 23–26.

[14] A. Basta, A. Blenk, M. Hoffmann, H. J. Morper, K. Hoffmann, and W. Kellerer, "SDN and NFV Dynamic Operation of LTE EPC Gateways for Time-varying Traffic Patterns," in *Proc. MONAMI*, Cham, 2015, pp. 63–76.

[15] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a cost optimal design for a 5g mobile core network based on sdn and nfv," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 14, no. 4, pp. 1061–1075, Dec. 2017.

[16] A. Basta, A. Blenk, S. Dudycz, A. Ludwig, and S. Schmid, "Efficient loop-free rerouting of multiple SDN flows," *IEEE/ACM Trans. on Netw. (ToN)*, vol. 26, no. 2, pp. 948–961, April 2018.

[17] C. Sieber, A. Basta, A. Blenk, and W. Kellerer, "Online resource mapping for SDN network hypervisors using machine learning," in *Proc. IEEE NetSoft Conf. and Workshops*, no. April, Jun. 2016, pp. 78–82.

[18] R. Durner, A. Blenk, and W. Kellerer, "Performance study of dynamic QoS management for OpenFlow-enabled SDN switches," in *Proc. IEEE IWQoS*, Jun. 2015, pp. 177–182.

[19] M. He, A. Basta, A. Blenk, and W. Kellerer, "How flexible is dynamic sdn control plane?" in *Proc. IEEE INFOCOM Workshop (INFOCOM WKSHPS)*, Apr. 2017, pp. 689–694.

[20] M. He, P. Kalmbach, A. Blenk, S. Schmid, and W. Kellerer, "Algorithm-Data Driven Optimization of Adaptive Communication Networks," in *Proc. IEEE ICNP Workshop ML&AI @ Network 2017*, Toronto, ON, Canada, Oct. 2017, pp. 1–6.

[21] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in sdn: Evaluation for dynamic flows," in *Proc. IEEE ICC*, Paris, France, May 2017, pp. 1–7.

[22] P. Kalmbach, J. Zerwas, A. Blenk, W. Kellerer, and S. Schmid, "Empowering self-driving networks," in *Proc. of the Afternoon Workshop on Self-Driving Networks*, ser. SelfDN 2018. ACM, 2018, pp. 8–14.

[23] W. Kellerer, A. Basta, and A. Blenk, "Using a flexibility measure for network design space analysis of SDN and NFV," in *Proc. IEEE INFOCOM Workshop*, Apr. 2016, pp. 423–428.

[24] W. Kellerer, A. Basta, P. Babarczi, A. Blenk, M. He, M. Kluegel, and A. Martínez Alba, "How to measure network flexibility? - a proposal for evaluating softwarized networks," *IEEE Commun. Mag.*, vol. PP, no. 99, pp. 2–8, 2018.

[25] S. Schwarzmann, A. Blenk, O. Dobrijevic, M. Jarschel, A. Hotho, T. Zinner, and F. Wamser, "Big-data helps sdn to improve application specific quality of service," in *Big Data and Software Defined Networks*, J. Taheri, Ed., Jan. 2018, pp. 433–455.

[26] J. Zerwas, P. Kalmbach, C. Fuerst, A. Ludwig, A. Blenk, W. Kellerer, and S. Schmid, "Ahab: Data-driven virtual cluster hunting," in *Proc. IFIP Networking*, Zurich, Switzerland, May 2018, pp. 1–9.

[27] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer, "Dynamic Application-Aware Resource Management Using Software-Defined Networking: Implementation Prospects and Challenges," in *Proc. IEEE NOMS*, Krakow, Poland, May 2014, pp. 1–6.

[28] T. Benson, A. Akella, and D. a. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM SIGCOMM IMC*, New York, New York, USA, Nov. 2010, pp. 267–280.

[29] J. Erman and K. Ramakrishnan, "Understanding the super-sized traffic of the super bowl," in *Proc. ACM SIGCOMM IMC*, 2013, pp. 353–360.

[30] J. L. Garcia-Dorado, A. Finamore, M. Mellia, M. Meo, and M. Munafo, "Characterization of ISP Traffic: Trends, User Habits, and Access Technology Impact," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 9, no. 2, pp. 142–155, Jun. 2012.

[31] V. Gehlen, A. Finamore, M. Mellia, and M. M. Munafò, *Uncovering the Big Players of the Web*, Berlin, Heidelberg, 2012, pp. 15–28.

[32] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[33] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[34] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. Mckeown, and G. Parulkar, "FlowVisor: A network virtualization layer," OpenFlow Consortium, Tech. Rep., 2009.

[35] R. Sherwood, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K.-K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown, M. Chan, G. Parulkar, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, and M. Kobayashi, "Carving research slices out of your production networks with OpenFlow," *ACM SIGCOMM CCR*, vol. 40, no. 1, pp. 129–130, Jan. 2010.

[36] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S.-H. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, "Network virtualization in multi-tenant datacenters," in *Proc. USENIX Symp. NSDI*, Apr. 2014, pp. 203–216.

[37] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. ACM Symp. on Operating Systems*, 2003, pp. 164–177.

[38] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. ACM Workshop on Hot Topics in Softw. Defined Netw.*, Aug. 2012, pp. 7–12.

[39] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. Maltz, "Latency inflation with MPLS-based traffic engineering," in *Proc. ACM SIGCOMM IMC*, Nov. 2011, pp. 463–472.

[40] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. ACM SIGCOMM Workshop on Internet Measurment (IMW)*, Nov. 2002, pp. 237–242.

[41] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *Proc. ACM SIGCOMM*, Aug. 2011, pp. 350–361.

[42] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure," in *Proc. ACM SIGCOMM*, New York, New York, USA, 2016, pp. 58–72.

[43] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," *Comm. of the ACM*, vol. 53, no. 4, p. 50, 2010.

[44] J. Zhu, Z. Jiang, and Z. Xiao, "Twinkle: A fast resource provisioning mechanism for internet services," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 802–810.

[45] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, Jun. 2016.

[46] A. Basta, A. Blenk, W. Kellerer, and S. Schmid, "Logically Isolated, Actually Unpredictable? Measuring Hypervisor Performance in Multi-Tenant SDNs," *Computer Research Repository (CoRR)*, pp. 1–7, 2017.

[47] perfbench. (2017) https://github.com/tum-lkn/perfbench.

[48] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, W. Snow, and G. Parulkar, "OpenVirteX: a network hypervisor," in *Proc. USENIX Open Netw. Summit (ONS)*, Santa Clara, CA, Mar. 2014.

[49] hyperflex. (2017) https://github.com/tum-lkn/HyperFLEX.

[50] J. L. Cohon, *Multiobjective programming and planning*, 2013.

[51] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons." *Proc. Natl. Acad. Sci. USA*, vol. 81, no. 10, pp. 3088–3092, May 1984.

[52] J. J. Hopfield and D. W. Tank, ""Neural" computations of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[53] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1–9.

[54] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal Virtual Network Embedding: Node-Link Formulation," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 10, no. 4, pp. 356–368, Dec. 2013.