Guest Editorial
Scalability Issues and Solutions for Software Defined Networks

Oliver Hohlfeld, James Kempf, Martin Reisslein, Stefan Schmid, and Nadir Shah

I. Introduction

A. Software-Defined Networks: A Success Story

Software Defined Networking [1] (in short SDN, which is also an acronym for Software-Defined Network), has emerged as a response to the limitations and complexities of traditional network architectures. At the heart of SDN lies the idea to consolidate the control over network devices into a logically centralized (software) controller separated from the data plane. The separation of the control plane and the data plane is realized via an open programming interface between the data plane switches and the SDN controller. The decoupling allows the control plane to evolve independently of the data plane, which enables faster innovation since software often exceeds hardware in innovation speed. Furthermore, logical centralization has the potential to simplify network operation and management by providing a single focal point where the consequences of management actions can be assessed, and possibly rejected if they would lead to some violation of operational constraints. OpenFlow, the standard SDN protocol today, is based on a simple match-action paradigm which results in great flexibilities, e.g., in terms of traffic engineering, definition of flows, as well as in-band network functionalities.

While originally proposed in an academic context, SDN has now achieved far-reaching impact in industry, with many companies, such as Google, Facebook, Yahoo, and Microsoft, promoting and adopting SDN through open standards development. Today, SDN is deployed in a wide range of network types: in enterprise and campus networks (where it originally started), in datacenter networks, in wide-area networks (for example, Google B4 [2]), as well as in Internet Exchange Points.

It is convenient to think of SDN in terms of its three main planes, which are illustrated in Fig. 1.

- **Data Plane:** The data plane consists of the network devices, such as the physical/virtual switches, routers, and access points. These data plane devices are managed by SDN controllers in the control plane.
- **Control Plane:** The SDN control plane consists of a set of software-based controllers that provide control functionality. Controllers have interfaces for communication with other controllers in the control plane, with network devices in the data plane, and with applications.
- **Application Plane:** End-user SDN applications interact with controllers to utilize an abstract view of the network for their internal decision making. Popular applications include traffic engineering and load balancing.

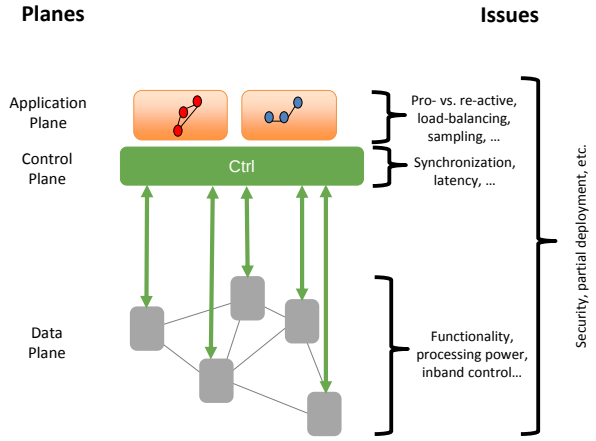

Fig. 1. Illustration of planes of an SDN and related scalability issues.

B. Scalability: The Next Frontier

Despite the large spectrum of SDN deployments, today's SDNs have in common that they are relatively small in scale. They are mostly limited to a small network or to a single administrative domain. There is a wide consensus that the next major challenge for SDN is to scale to massive numbers of routers and switches, as well as to address multi-domain networks.

We first have to note that while the centralized perspective offered by SDN is attractive, it was always meant to be *logically* centralized only, but physically *distributed*. Furthermore, there have already been a number of scalability related improvements in OpenFlow specifications over the last few years, for example the introduction of a group table mechanism (OpenFlow version 1.1) which allows multiple flow table entries to point to (reference) the same group identifier: the

O. Hohlfeld is with the Network Architectures Group, COMSYS, RWTH Aachen University, Germany (e-mail: oliver@comsys.rwth-aachen.de).

J. Kempf is with Ericsson Research (e-mail: james.kempf@ericsson.com).

M. Reisslein is with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706, USA (e-mail: reisslein@asu.edu), phone 480-965-8593.

S. Schmid is with the Faculty of Computer Science, University of Vienna, Austria (e-mail: stefan_schmid@univie.ac.at).

N. Shah is with the Department of Computer Science, Comsats Institute of Information Technology, Wah Cantt, Pakistan, 47040 (e-mail: nadir-shah82@gmail.com).

controller only needs to update the referenced group table entry action instead of updating the action of all flow table entries. OpenFlow version 1.2 also allows to change the controller, enabling a switch to establish communication with multiple controllers in parallel.

Scalability involves generally a wide range of aspects and a system is only as scalable as its least scalable component, i.e., the component that first becomes a bottleneck as the network size scales up. The research addressing SDN scalability can be classified into the following main categories: data plane, control plane, applications, and cross-cutting issues.

1) Data Plane Scalability: Data plane scalability mainly depends on the processing power, memory/buffer capacity, and software implementation of the data plane devices [3]. One fundamental scalability question related to the data plane regards which functionality to include in the data plane in order to mitigate the controller load. A specialized data plane may be used to keep more traffic in the data plane, toward the goal of a more scalable and efficient control plane. Another scalability challenge arises in case of in-band control of data plane devices, where the control traffic shares paths (and competes for resources) with the data plane traffic, and where a reliable management of the data plane needs to be ensured.

2) Control Plane Scalability: The majority of the studies on SDN scalability addresses the control plane. The most prominent SDN control plane performance metrics are throughput (number of flow requests handled per second) and (flow setup) latency. A first set of scalability issues can result from the limited amount of requests that can be handled by a given controller per time. Distributed, flat, or hierarchical control plane designs can mitigate the controller bottleneck. However, such designs require inter-controller synchronization and data distribution among controller replicas. Further scalability issues arise from the separation of the control and data planes, as now, network devices are managed by a *remote* controlling mechanism. Indeed, the controller-switch communication delay introduces a delay, e.g., for flow setup. The situation is particularly problematic if the communication network between controllers and switches can be congested (e.g., in case of in-band management). In general, in order to reduce control latency in wide-area networks, data plane events should be handled close to where they occur, raising the question of where to place controllers.

3) Application Scalability: SDN applications themselves need to be scalable and support fine-grained and optimized resource utilization in wide-area networks. Load balancing was one of the first applications envisioned for SDN. One example technique to make load balancing applications scalable is to use wildcard-based rules to perform proactive load balancing [1]. Such wildcard-based rules aggregate client requests, e.g., based on IP prefix ranges.

4) Cross-cutting Scalability Issues: SDN will likely be deployed incrementally in large networks, which raises many other scalability related questions. For instance, which switches should be upgraded first and how can an SDN operation be emulated with only a partial deployment of SDN devices [4].

Further cross-cutting scalability issues arise in the context of software-defined wireless networks. The wireless transmission medium introduces additional opportunities for control and optimization, i.e., presents additional tuning knobs that can be set with SDN control. However, fine-grained wireless transmission control can potentially give rise to novel bottlenecks in SDNs.

The larger the SDN, the more attractive it may become as a target of attacks. As an SDN is only as secure as its individual planes, strong yet scalable security mechanisms (e.g., for monitoring, detecting, and recovering from attacks) are required for the data plane, the control plane, and the application plane.

II. Summary of Research Contributions in this Special Issue

This Special Issue (SI) presents original state-of-the-art research studies on scalability issues and solutions for SDNs. The relevance of and interest in this topic area is reflected by the large number of submissions: we received a total of 70 submissions from authors distributed geographically over 21 countries and all 6 continents (19 from China, 15 from USA, 3 each from Germany, Korea, Canada, France, and Tunisia, 2 each from Japan, New Zealand, UK, Turkey, Poland, Spain, and Taiwan, and 1 each from Hong Kong, Hungary, Brazil, Montenegro, Italy, Australia, and India). All papers received at least three reviews and accepted papers went through at least one revision round. Out of the 70 papers, we eventually accepted 17 papers (acceptance ratio 24%).

The articles contained in this SI cover data plane, control plane, and application plane scalability issues and related solutions. The articles report on recent developments in architectural specifications, protocols, and application designs for achieving scalability in SDN. The papers in this SI employ a mixture of experimental, conceptual, and theoretical approaches to examine SDN scalability. The following subsections summarize the papers contained in this SI.

A. Data Plane

Patra et al. in *Towards a Sweet Spot of Data plane Programmability, Portability and Performance: On the Scalability of Multi-Architecture P4 Pipelines* examine the performance, portability, and scalability of the SDN data plane. Programming Protocol-Independent Packet Processors (P4) provide a high-level programming language for fine-grained programmability of the SDN data plane. Patra et al. argue that the P4 adoption is hindered by a lack of open source, protocol independent programmable data plane components. Patra et al. present a Multi-Architecture Compiler System for Abstract Data planes (MACSAD) concept. MACSAD builds on application programming interfaces (APIs) from the OpenDataPlane (ODP) project to provide low-level hardware and software cross-platform programmability. Patra et al. identify three critical evaluation measures, namely performance, portability, and scalability. They comprehensively evaluate the trade-offs between these three measures for a wide range of

packet sizes, pipeline complexity levels, and numbers of cores across different platforms (e.g., x86 and ARM).

Oudin et al. in *OFLOPS-SUME and the art of switch characterization* introduce a novel hardware/software co-design for SDN switch characterization at 40 Gbps and beyond. OFLOPS-SUME integrates the software based open framework for OpenFlow Switch Evaluation (OFLOPS) with the hardware based Open Source Network Tester (OSNT). The resulting OFLOPS-SUME can evaluate any hardware and software device with an OpenFlow control plane, including P4 programmable data plane devices. Oudin et al. conduct an extensive evaluation study of both software and hardware switches and identify the implications of different switch profiles on the overall network scalability and performance. Oudin et al. also release OFLOPS-SUME as an open source tool so as to facilitate the further refinement of the OFLOPS-SUME switch characterization tool and to support widespread reproducibility of SDN switch characterizations.

The next-generation 5G core will depend on a programmable data plane to support mobile edge computing, network slicing, and network function virtualization at scale. In order to support telcos and vendors to make the transition from the legacy mobile core to an SDN-based 5G-capable data plane, Lévai et al. in *The Price for Programmability in the Software Data Plane: The Vendor Perspective* evaluate the performance and scalability of several programmable software switches for representative scenarios taken from real 5G deployments. In particular, Lévai et al. first introduce a taxonomy for data plane scalability together with a novel data-plane scalability benchmarking tool called TIPSY for Telco pIPeline benchmarking SYstem. Then, a set of ten standard telco pipelines is defined for benchmarking. Programmable software data planes based on eight different types of SDN programmable switches are then benchmarked. The evaluations indicate that most programmable switches result in significantly reduced performance, i.e., requiring significantly higher investments in hardware and energy (by up to roughly an order of magnitude) than conventional statically configured systems, i.e., there is a very high price to pay for the flexible configurability and management through programmable switches. Only one proprietary programmable switch has demonstrated the potential to match the performance of conventional systems.

Khalili et al. in *Flow setup latency in SDN networks* examine a path aggregation strategy to reduce the flow setup latency as SDN networks scale to large numbers of switches. Khalili et al. first measure flow setup latencies in conventional SDN networks and discover that the 99 percentiles of the latencies are on the order of half a second. In order to address these long latencies, Khalili et al. propose to pre-configure pipes that interconnect any pair of edge switches. The individual flows are processed at the ingress edge switches, from where they are forwarded through pre-configured fabric pipes to the egress edge switches. This approach reduces the number of switches that need to be configured for setting up a flow to typically two switches, decreasing the 99 percentile latencies to less than 10 milliseconds.

Xiao and Krunz in *Dynamic Network Slicing for Scalable Fog Computing Systems with Energy Harvesting* investigate SDN-based dynamic network slicing. Specifically, network slicing for a prescribed region of fog nodes is controlled by a regional orchestrator. The orchestrator coordinates with the regions fog nodes to dynamically distribute the resources of the network slices according to the local service demands and availability of harvested energy. Xiao and Krunz model the dynamic network slicing and the resource allocation problem as a stochastic overlapping coalition-formation game. The model reveals an increase in the overall computing capacity if fog nodes consider a belief function about the unknown states and private information of the other fog nodes. Xiao and Krunz optimize the dynamic network slicing through a belief-state partially observable Markov decision process. Numerical evaluations based on data from 400 base station locations in a real cellular network illustrate that cooperation of each fog node with its closest neighbor nearly doubles the processed workload.

Allybokus et al. in *Multi-Path Alpha-Fair Resource Allocation at Scale in Distributed Software Defined Networks* address the problem of fair bandwidth sharing for good network performance. In particular, the authors consider allocation problems for the important case that flows evolve over time. The authors propose a distributed algorithm to solve the multi-path fair resource allocation problem in a distributed SDN control architecture, overcoming the challenges of standard primal-dual decomposition methods. The authors then demonstrate scalability of their approach, on large instances with hundreds of nodes and thousands of requests and paths.

Bruyere et al. in *Rethinking IXPs Architecture in the Age of SDN* propose to improve the scalability of the SDN control plane by handling the control traffic directly within the data plane. This approach is realized in the Umbrella switching fabric architecture and management approach. The Umbrella management approach improves the overall robustness by limiting the control plane dependency and is suitable for existing Internet eXchange Point (IXP) topologies. The scalability of Umbrella is demonstrated through the successful deployment at two commercial IXPs. Umbrella can be seen as a first step towards SDN architectures that are less dependent on the control plane and thus are better scalable. In such future architectures, the data plane components support the controller in its role of an intelligent supervisor, rather than as an active and critical decision point.

B. Control Plane

Yan et al. in *BigMaC: Reactive Network-wide Policy Caching for SDN Policy Enforcement* address control and data plane scalability by proposing a reactive policy enforcement framework. BigMaC's main contribution is to provide a policy-consistency guarantee (i.e., packets should receive the same actions in the big switch abstraction and in the data plane switches with cached entries) across the network. The model presents layered views of the network with a big switch abstraction, a logical network, and a physical network. Yan et al. propose a network-wide "bucket"-based policy mapping and caching mechanism which guarantees policy-consistency,

optimizes the flow table usage of physical network devices, and reduces the churn incurred by policy updates. Trace-driven simulations indicate that BigMaC achieves better performance than traditional schemes, in particular, BigMaC saves table space and reduces the update complexity.

Görkemli et al. in *Dynamic Control Plane for SDN at Scale* have been motivated by the scalability and reliability requirements of wide area networks and 5G core networks. In particular, the paper is concerned about the possibility of bottlenecks due to the controller CPU or the throughput of the in-band control channels. The authors propose a programmable distributed control plane architecture based on a dynamically managed in-band control network. Towards this end, the authors introduce a "control flow table" to manage in-band control flows, enabling for instance the offloading of congested controllers and congested in-band control channels.

Sakic and Kellerer in *Impact of Adaptive Consistency on Distributed SDN Applications: An Empirical Study* examine the scalability of the SDN control plane where state is replicated, e.g., for fault-tolerance. Sakic and Kellerer explore the effects of deployed consistency models on scalability and correctness, comparing strong and eventual consistency, and make a case for a novel adaptive consistency model. Sakic and Kellerer show how an adaptive consistency model offers scalability benefits in terms of the total request handling throughput and response time, in contrast to the strong consistency model. They also outline how the adaptive consistency model can provide correctness semantics, that are unachievable with the eventual consistency model. The paper further reports on an emulated testbed with a load balancer controller application.

Lyu et al. in *Multi-timescale Decentralized Online Orchestration of Software-Defined Networks* enhance SDN scalability with a hierarchical network of controllers, whereby controllers further down in the hierarchy can be switched off at different times depending on traffic conditions. Each switch in the network is associated with a controller at the first level, and if the controller is switched off, it forwards PACKET_IN messages from switches to the next level up. Lyu et al. develop an analytical formulation of a distributed online optimization problem which allows the control plane to determine the optimal set of active controllers. They test the algorithm in a MATLAB simulation platform with 320 switches and 14 controllers arranged in three levels: two root controllers, four intermediate controllers, and eight local controllers. They simulate three orchestration approaches: static orchestration, real-time dynamic orchestration, where the controller activation, request processing, and dispatching are carried out instantaneously at every time slot, and T-slot dynamic orchestration, where the request processing and dispatching are carried out per slot and the controller activation is carried out every T slots, with $T = 10$. The real-time orchestration achieves the lowest system cost, saving 73% over the static orchestration cost. The $T = 10$ slot orchestration approach achieves slightly less optimal performance than the real-time approach.

C. Scalable SDN Applications and Use Cases

Cheng and Jia in *Compressive Traffic Monitoring in Hybrid SDN* investigate scalable load estimation, which can inform traffic engineering (an important application of SDN control). In particular, Cheng and Jia propose a novel compressive traffic monitoring method for the accurate real-time collection of load information of all links. The focus is on hybrid SDN networks and the main idea is to judiciously place a small number of SDN routers such that controllers only need to collect the load information of a small subset of important links. The loads of the other links are then estimated.

Moradi et al. in *Dragon: Scalable, Flexible and Efficient Traffic Engineering in Software Defined ISP Networks* describe the Dragon SDN-based traffic engineering system for Internet Service Provider (ISP) networks that scales to large networks. To address the scalability challenge, Dragon consists of hierarchical and recursive traffic engineering (TE) algorithms and mechanisms that divide flow optimization problems into subtasks and execute them in parallel. This approach finds an approximate solution much quicker than a full optimization problem. Dragon further enables ISPs to express diverse objectives for different parts of their network. Dragon also has optimizations to reduce the size of packets (avoiding excessive headers) and the ternary content-addressable memory (TCAM) size. Dragon has been evaluated for several network topologies.

Fu et al. in *Taming the Wild: A Scalable Anycast-based CDN Architecture (T-SAC)* utilize SDN to improve the scalability of anycast-based content distribution networks (CDNs) by proposing to use OpenFlow and network function virtualization (NFV) to complement current domain name system (DNS)-based redirection schemes. T-SAC achieves fine-grained control on redirections within the CDN through load-aware algorithms and setting a re-direction bit in the status/health check messages. T-SAC aims at fostering the collaboration of CDNs and ISPs by enabling more flexible routing decisions than a standard anycast architecture. The network itself measures the load imposed on CDN nodes and takes according redirection decisions. At the same time, the CDN may influence the routing decision by signaling that certain CDN nodes are not capable of serving additional connections. The benefits of the T-SAC architecture are demonstrated by a real Amazon Web Services (AWS) based implementation.

Uddin et al. in *SDN-based Multi-Protocol Edge Switching for IoT Service Automation* propose an architecture called Muppet, for multiprotocol at the edge for IoT. The Muppet architecture consists of a collection of P4 Muppet switches at the edge that handle peer-to-peer non-IP traffic by tunneling over IP to other Muppet switches on the edge near the destination devices. This allows the IoT network to expand beyond a simple peer-to-peer network by using the IP backbone. The Muppet switches are programmed from a P4 SDN controller; the controller can install policy rules that allow traffic to be filtered depending on content. The Muppet architecture is divided into three planes: a management plane which onboards devices having specific link layer protocols, a data plane which supports packet inspection and processing using match-action rules, and a control plane that allows the P4 controller to program the switches. Uddin et al. demonstrate the Muppet architecture by supporting two different non-IP protocols, namely Bluetooth Low Energy (BLE) and Zigbee.

The authors then describe a prototype implementation which reduces the power utilization and increases the throughput compared to alternative systems (peer to peer with BLE, cloud with WiFi, and cloud with LTE). Muppet achieves these performance improvements by allowing the switches to filter device readings that do not match a specific policy (e.g., temperature readings that are above a threshold). They also tested the scalability using virtual BLE adaptors and found that the CPU usage of Muppet switches scales linearly with the number of ports, up to 1000 ports.

Zhang and Zhu in *Scalable Virtualization and Offloading Based Software-Defined Architecture for Heterogeneous Statistical QoS Provisioning Over 5G Multimedia Mobile Wireless Networks* propose an SDN architecture for offloading multimedia traffic to efficiently support multimedia traffic in 5G networks. The architecture consists of three virtual networks: a virtual network without offloading, a virtual network with WiFi offloading, and a virtual network with peer-to-peer offloading. The architecture achieves statistical QoS provisioning by different means on the different virtual networks. On the virtual networks without offloading, the authors analytically derive optimal wireless resource allocation to maximize spectral efficiency of aggregate effective capacity per hertz. On virtual networks with WiFi offloading, an optimal power allocation scheme is developed to maximize the aggregate effective capacity. On virtual networks with peer-to-peer offloading, the performance improvements in terms of throughput, data content, and downloading delay are modelled. An SDN control plane makes the decision about what traffic to offload based on the characteristics of the traffic. Zhang and Zhu also estimate the scalability improvement of the overall integrated network. The advantages of the architecture are demonstrated through an analytical comparison with a standard base station cellular deployment.

Fawcett et al. in *TENNISON: A Distributed SDN Framework for Scalable Network Security* describe a system for security monitoring and control based on a three layer architecture. At the bottom, in the Collection layer, existing tools collect data at L1 (sFlowRT), L2 (Open Network Operating System, ONOS), and L3 (Snort and Bro) to provide control and monitoring to higher layers. At the next layer, the Coordination layer, the Tennison controller provides coordination between different security applications. At the top layer, the Application layer, various security applications run and coordinate their activity southbound through the Tennison controller API. The Tennison controller design scales readily to multiple distributed instances so as to match the network topology. Fawcett et al. present emulation results from a Mininet simulation with 350 nodes connected to 19 partially connected switches, which is representative of a large-size business network, for four attack scenarios: Denial of Service (DoS), Distributed DOS (DDoS), scanning, and intrusion. They found that Tennison was able to detect an attack in less than 6 seconds, with the maximum amount of time needed for DDoS.

III. The Road Ahead: Open Research Challenges

While the papers contained in this SI have achieved significant advances in making SDNs more scalable, there are still many important research challenges that need to be addressed to make large-scale SDNs a reality. We provide a perspective of the main open research challenges towards scalable SDNs in this section. We first outline the main open research challenges in the data plane, the control plane, and the applications plane. We then outline several open research challenges that cut across these different planes and require a holistic approach to scalability.

A. Data Plane

1) Flow Tables: SDN switches are kept very simple by implementing only elementary data plane functionalities. The simple SDN switch design poses the challenge of efficiently using the limited SDN switch resources. For instance, how can the limited switch TCAM memory be used efficiently to store increasing numbers of active flows in the TCAM based switch flow table? How can the matching of an incoming packet with the patterns of large numbers of flow entries stored in the switch flow table be sped up? How can flow updates at the switches be performed efficiently to avoid inconsistencies? How can packets waiting for the controller commands be efficiently buffered at the switches? While a few approaches have been proposed to address these challenges, see e.g., [5], [6], the scalability characteristics of flow table management require extensive future research.

2) Adding Capabilities to Data Plane: There is an emerging trend to offload more functionality to the data plane, e.g., through so-called smart network interface controllers (NICs) that take over some of the control plane functions. The "smarter" data plane strives to offload the controller so as to make the controller more scalable. A smarter data plane can increase scalability also along other dimensions and positively affect applications [7], [8]. Aligned with this trend, and to give another example, there are studies evaluating under which conditions NICs and switches are suitable to work as CPU co-processors for neural networks, accelerating artificial intelligence AI [9].

3) Consistent Data Plane Updates: The study [10] has addressed the problem of efficiently updating the flow tables at the switches to avoid inconsistencies. However, for large networks with frequent updates, the proposed protocol does not perform in real time and requires extensive computation resources and network bandwidth. Thus, the consistent updating of the data plane in a scalable manner is still an open research challenge.

4) Scalable Resilience Mechanism in Data Plane: Links and nodes can fail in a network, affecting flows for critical applications [11]. To provide resilience against such failures, traditional networks employ distributed protocols. The logically centralized controller in SDNs can facilitate the efficient failure handling. The failure handing in SDNs poses unique challenges and several approaches have been proposed, e.g., [11]. In order to provide resilience against failures, the existing approaches, install multiple paths for a given flow at the switches. The computational processing capacity and flow table memory of switches limit the scalability of the approach of installing multiple paths for a given flow. An open challenge

is to examine approaches that compute the reliability levels of links and nodes, and then install the appropriate numbers of multiple paths based on the reliability levels. More specifically, if a path is highly stable, then we should not install multiple paths for a flow at the switches along the path. Otherwise, we should install a number of multiple flow paths that is appropriate for compensating for the reliability levels of the paths.

B. Control Plane

A single controller can limit the scalability of an SDN. Multi-controller architectures have recently been proposed to increase the scalability of SDNs. However, there are several open challenges that still need to be addressed for multi-controller architectures [12]. One main open challenge is the optimal arrangement of the controllers. Generally, multiple controllers can be arranged in different ways, such as isolated (in which a controller manages its own domain without communicating with other controllers), flat (in which multiple controllers are arranged in a flat structure and the controllers coordinate with each other to control and manage the network), and hierarchical (in which the controllers are arranged in layers and each controller has different roles, for example, local controller vs. global controller).

Another open challenge is the optimal controller placement. The controllers should be placed to support efficient network operation. The controllers can be placed according to different operational objectives, e.g., each host should have the nearest controller within a prescribed maximum distance, or the overall flow setup latency should be minimized, or each controller should handle the same number of flows so as to balance the controller loads, or various combinations of these objectives.

The existing approaches for distributed SDN control, which have been surveyed in [12], limit some of the scalability aspects. For example, different links and switches have different reliability levels [11]. In order to address the reliability issue, the study [13] has advocated to place multiple controllers in the network using an in-band control plane. The in-band control plane conducts the communication between the switches and the controller over the communication links used for communication among the switches [14]. In particular, the study [13] has advocated to connect the switches to the controllers so as to fairly balance the loads on the controllers, while the controllers back each other up in case of failures. Future research needs to consider the reliability levels of the paths from the switches to the controllers as well as the paths interconnecting the controllers in the switch to controller assignments. The reliability levels of a path will depend on the reliability of the bottleneck link of the path, i.e., the link with the lowest reliability level. More broadly, future research needs to ensure that the control plane does not become a performance bottleneck, nor a source for reliability problems as SDNs scale up in size and number of handled flows. Moreover, the control plane needs to efficiently and reliably support network virtualization, which requires hypervisors that intercept the data plane to controller traffic.

C. Scalable SDN Applications and Use Cases

1) Scalable Approaches for Auto Network Debugging: Recently, many approaches have been proposed for debugging and verification of the network configurations and operations, such as efficient policy enforcement (EPE) [15] and RuleScope [16]. These approaches have major scalability problems: when the network scales, e.g., in term of number of faulty rules [16] or number of switches and flow rules [15], the execution time increases exponentially. Thus, as the network sizes scale up, it becomes increasingly harder to verify the network operations in real time. Therefore, the scalability of auto network debugging approaches is an important open research challenge.

2) Network Updates: Network maintenance and upgrades are vital tasks that require care and attention. Upgrades or maintenance of existing device hardware and software require temporarily the shifting of network traffic to other devices. Nevertheless, the overall network traffic service level should be maintained and congestion should be avoided. A range of mechanisms, e.g., [17], [18], have been proposed for achieving these objectives in SDNs. However, these approaches suffer from scalability issues: when the network size increases in terms of the number of switches, the size of the traffic matrix, or the number flow rules, then the computation times of these approaches increase linearly. This can lead to network performance degradations, e.g., long computation times will cause congestion during the execution of the computations. Scalable network update mechanisms and corresponding resource allocation and dimensioning approaches are needed to ensure uninterrupted network service during upgrades and maintenance need to be researched.

D. Cross-Cutting Challenges

1) Hybrid SDNs: Despite the benefits of SDN, many organizations hesitate to adopt SDN in practice due to a variety of reasons [4]. One main reason is that organizations often hesitate to invest large funds at once to completely replace traditional networks by SDNs. In order to reap the benefits of SDN while avoiding these concerns, researchers have proposed a new network architecture, referred to as a hybrid SDN. Hybrid SDNs enable the organizations to incrementally replace their traditional network devices with SDN devices. Due their unique architecture, hybrid SDNs pose many scalability related research challenges.

These research challenges span all SDN planes. For instance, one data plane challenge is to optimize the locations of the SDN switches. An example control plane challenge is to design a scalable controller that can effectively control a large traditional network through a small set SDN devices. Moreover, scalable protocols for communication among SDN devices and traditional network devices are an open research challenge. For example, Cheng and Jia in “Compressive Traffic Monitoring in Hybrid SDN” in this SI achieve scalability by collecting the load levels of a small subset of important links and then use these load samples to infer the load levels of the remaining links. However, the problem of computing the subset of important links is NP-complete. Moreover, the

problem of identifying the traditional routers to be replaced by SDN switches such that all important links are SDN-based is NP-hard.

A related open problem is the hybrid SDN control for large-scale hybrid SDNs. The recent Magento study [19] has proposed an SDN control mechanism to control traditional switches through SDN switches in a hybrid SDN. However, the performance evaluation of the Magneto control mechanism revealed that for a given fraction of SDN switches, a lower percentage of traditional switches is controlled by the SDN control mechanism in large-sized network topologies as compared to small-size and medium-size typologies [19, Fig. 7(c)]. Future research needs to develop effective hybrid SDN control mechanisms that scale well for large network topologies.

2) Plane Interfaces: The standardized SDN interfaces between the various planes are critical for the success of SDN. The so-called south-bound interface (application programming interface, API), which is formally referred to as data-controller plane interface (D-CPI), interconnects the control plane with the data plane, while the so-called north-bound interface, formally referred to as application-controller plane interface (A-CPI), interconnects the control plane with the applications plane. Moreover, the intermediate controller plane interface (I-CPI) interconnects the SDN controllers of different network domains. Clearly, highly scalable SDNs requires highly "scalable interfaces" that support efficient communication between the planes. More specifically, the interfaces need to be designed to efficiently handle growing flow numbers and network sizes. Growing network sizes, for instance, will likely require complex distributed control planes, which in turn complicate the south-bound interface. Accordingly, an interesting open question is whether today's interfaces are indeed future proof, or whether improved programming interfaces need to be designed.

3) Scalable Wireless SDN Networks: Recently, SDN has been used in wireless networks for enhancing a wide range of wireless network functions, such as traffic offloading, making the wireless data plane more programmable, and managing node mobility more efficiently [20]–[22]. Some studies have sought to make these wireless network enhancements scalable for large networks. For example, Aeroflux [21] has introduced two types of controllers, namely local controllers and global controllers. The local controller handles the local wireless network events while the globular controller handles the global events so as to improve the overall wireless network scalability. As another example, the SDN based smart gateway (SmGW) [22] aggregates a large number of local wireless access nodes, e.g., numerous femto call base stations, to simplify the backhaul of the wireless traffic to and from the Internet at large.

Despite these and other recent advances towards scalable wireless SDNs, vast open research challenges remain. For instance, in the context of the emerging 5G wireless standard, thousands or even millions of Internet of Things (IoT) devices require connectivity at a fraction of the cost of todays mobile devices. The question of which functionality to include in the data plane is particularly interesting for wireless SDNs due to the additional "tuning knobs" related to, e.g., power and rate control, as well as the inherent node mobility and changing wireless channel conditions. Additional challenges arise in multi-tenant wireless SDNs that require scalable isolation of the numerous wireless network slices and the abstraction of the wireless network characteristics for the individual tenant controllers.

4) Security: A SDN needs efficient and powerful security mechanisms to avoid security vulnerabilities across the data, control, and application planes. A number of security mechanisms have been proposed for SDNs, as surveyed in [23]. A security mechanism generally involves three phases, namely monitoring the network (this generates extra traffic both at the control and data planes), detecting the security breach (this takes some time for algorithm execution), and the recovery (once the attack is detected, this phase incurs both time delay and traffic overhead by taking the proper counter measures against the detected attack). The scalability of the existing security mechanisms is a big concern as the number of hosts, switches, controllers, flows, and attackers increases. The existing approaches typically attempt to achieve the scalability by improving the performance of an individual phase. For example, Fawcett et al. in *TENNISON: A Distributed SDN Framework for Scalable Network Security* in this SI reduce the traffic overhead and the execution time of the monitoring phase. Similarly, the Athena approach [24] focuses on avoiding security vulnerabilities in the data plane. The main open research challenge is to develop holistic security approaches that improve the performance of all phases across all planes. These holistic approaches should reduce the execution time and increase the accuracy for increasing numbers of controllers and flows.

IV. Concluding Remarks

We are indebted to the authors that submitted papers for review for this SI. We are grateful to all the anonymous reviewers. This SI would not have been possible without the timely thorough reviews which have helped greatly to further refine and improve the paper in the revision round. We thank Muriel Médard and Raouf Boutaba, the former and present Editor-in-Chief of the IEEE Journal on Selected Areas in Communications for their helpful guidance. A special thank you goes to Laurel Greenidge and Janine Bruttin, former and current Executive Editor of IEEE JSAC, who helped with countless logistical and procedural issues during the paper review and SI production process. We are also grateful to Prof. Moshe Zukerman, IEEE JSAC Senior Editor, for shepherding this SI.

We hope you enjoy reading the articles contained in this SI and will be inspired by the reported research studies. We look forward to seeing your future contribution to scalable SDNs!

Oliver Hohlfeld, James Kempf, Martin Reisslein, Stefan Schmid, and Nadir Shah

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, and et al., "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, Oct. 2013.

[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[4] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surv. & Tut., in print*, 2018.

[5] T. Cheng, K. Wang, L.-C. Wang, and C.-W. Lee, "An in-switch rule caching and replacement algorithm in software defined networks," in *Proc. IEEE Int. Conf. on Communications (ICC)*, 2018, pp. 1–6.

[6] S. Yingchareonthawornchai, J. Daly, A. X. Liu, and E. Torng, "A sorted-partitioning approach to fast and scalable dynamic packet classification," *IEEE/ACM Trans. on Netw.*, vol. 26, no. 4, pp. 1907–1920, Aug. 2018.

[7] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, "In-network computation is a dumb idea whose time has come," in *Proc. ACM HotNets*, 2017, pp. 150–156.

[8] L. Schiff, M. Borokhovich, and S. Schmid, "Reclaiming the brain: Useful OpenFlow functions in the data plane," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2014, pp. 1–7.

[9] D. Sanvito, G. Siracusano, and R. Bifulco, "Can the network be the AI accelerator?" in *Proc. ACM SIGCOMM Workshop on In-Network Computing*, 2018, pp. 20–25.

[10] J. H. Han, P. Mundkur, C. Rotsos, G. Antichi, N. Dave, A. W. Moore, and P. G. Neumann, "Blueswitch: Enabling provably consistent configuration of network switches," in *Proc. ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS)*, 2015, pp. 17–27.

[11] P. Thorat, S. Jeon, and H. Choo, "Enhanced local detouring mechanisms for rapid and lightweight failure recovery in OpenFlow networks," *Computer Communications*, vol. 108, pp. 78–93, Aug. 2017.

[12] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, First Qu. 2017.

[13] D. M. Mattos, O. C. M. Duarte, and G. Pujolle, "A resilient distributed controller for software defined networking," in *Proc. IEEE Int. Conf. on Communications (ICC)*, 2016, pp. 1–6.

[14] M. Canini, I. Salem, L. Schiff, E. M. Schiller, and S. Schmid, "A self-organizing distributed and in-band sdn control plane," in *Proc. IEEE Int. Conf. on Distr. Computing Sys. (ICDCS)*, 2017, pp. 2656–2657.

[15] M. Hussain and N. Shah, "Automatic rule installation in case of policy change in software defined networks," *Telecommunication Systems*, vol. 68, no. 3, pp. 461–477, Jul. 2018.

[16] X. Wen, K. Bu, B. Yang, Y. Chen, L. E. Li, X. Chen, J. Yang, and X. Leng, "RuleScope: Inspecting forwarding faults for software-defined networking," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2347–2360, Aug. 2017.

[17] Y. Yuan, S. Chandrasekaran, L. Jia, and V. Sekar, "Efficient and correct test scheduling for ensembles of network policies," in *Proc. USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, Apr. 2018, pp. 437–452.

[18] A. El-Hassany, P. Tsankov, L. Vanbever, and M. Vechev, "NetComplete: Practical network-wide configuration synthesis with autocompletion," in *Proc. USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, Apr. 2018, pp. 579–594.

[19] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, "Magneto: Unified fine-grained path control in legacy and OpenFlow hybrid networks," in *Proc. ACM Symp. on SDN Res.*, 2017, pp. 75–87.

[20] X. Costa-Perez, A. Garcia-Saavedra, X. Li, T. Deiss, A. de la Oliva, A. di Giglio, P. Iovanna, and A. Moored, "5G-Crosshaul: An SDN/NFV integrated fronthaul/backhaul transport network architecture," *IEEE Wireless Commun.*, vol. 24, no. 1, pp. 38–45, Feb. 2017.

[21] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: A survey," *ACM Computing Surveys*, vol. 47, no. 2, pp. 27:1–27:11, Jan. 2015.

[22] A. Thyagaturu, Y. Dashti, and M. Reisslein, "SDN based smart gateways (Sm-GWs) for multi-operator small cell network management," *IEEE Trans. Netw. Serv. Managm.*, vol. 13, no. 4, pp. 740–753, Dec. 2016.

[23] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1701–1725, Third Qu. 2017.

[24] S. Lee, J. Kim, S. Shin, P. Porras, and V. Yegneswaran, "Athena: A framework for scalable anomaly detection in software-defined networks," in *Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*, 2017, pp. 249–260.

PLACE PHOTO HERE

Oliver Hohlfeld leads the network architectures group at RWTH Aachen University. He received the Ph.D. in computer science from TU Berlin in 2013. From 2008-2013 he was with TU Berlin / Deutsche Telekom Innovation Laboratories. His research focuses on empirically understanding networked systems through large-scale measurements and subsequently optimizing their user experiences.

PLACE PHOTO HERE

James Kempf (SM'18) graduated with a Ph.D. in Systems and Industrial Engineering, Computer Science minor, from University of Arizona, Tucson, in 1984 and promptly went to work in Silicon Valley. Prior to his current position, he spent 3 years at HP, 13 years at Sun Microsystems, primarily in research, and 8 years at Docomo Labs USA as a Research Fellow. Dr. Kempf worked for 10 years in IETF, was chair of 3 working groups involved in developing standards for the mobile and wireless Internet and was a member of the Internet Architecture Board for two years. He is the holder of 26 patents, and the author of many technical papers and 3 books, the latest of which, *Wireless Internet Security: Architecture and Protocols* was published by Cambridge University Press in 2008. Since 2008, Dr. Kempf works as a Principal Researcher at Ericsson Research in Silicon Valley on Software Defined Networking (SDN), Network Function Virtualization (NFV), and Cloud Computing.

PLACE PHOTO HERE

Martin Reisslein (S'96-M'98-SM'03-F'14) is a Professor in the School of Electrical, Computer, and Energy Engineering at Arizona State University (ASU), Tempe. He received the Ph.D. in systems engineering from the University of Pennsylvania, Philadelphia, in 1998. He currently serves as Associate Editor for the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Education*, and *IEEE Access*, as well as *Computer Networks*. He is Associate Editor-in-Chief of the *IEEE Communications Surveys & Tutorials* and Co-Editor-in-Chief of *Optical Switching and Networking*. He chairs the steering committee of the *IEEE Transactions on Multimedia*.

PLACE PHOTO HERE

Stefan Schmid is a Professor in Computer Science at University of Vienna, Austria. He received the Ph.D. (Dr.sc.) from ETH Zurich, Switzerland, in 2008. He currently serves as Editor of the Distributed Computing Column of the *Bulletin of the EATCS* and as Associate Editor for the *IEEE Transactions on Network and Service Management (TNSM)*. Stefan Schmid received the IEEE Communications Society ITC Early Career Award 2016.

PLACE PHOTO HERE

Nadir Shah received the B.Sc. and M.Sc. degrees from Peshawar University, Peshawar, Pakistan, in 2002 and 2005, the M.S. degree from International Islamic University, Islamabad, Pakistan, in 2007, all in computer science, and the Ph.D. degree from the Sino-German Joint Software Institute, Beihang University, Beijing, China. He was a Lecturer with the Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan, from 2007 to 2008. He is currently an Associate Professor with the COMSATS University Islamabad, Wah Campus. His current research interests include computer networks, distributed systems, and network security. He is serving in the editorial board of IEEE Softwarizations, AHWSN and MJCS.