SDN in Wide-Area Networks: A Survey

Oliver Michel, Eric Keller
University of Colorado Boulder
{oliver.michel, eric.keller}@colorado.edu

***Abstract*—Over the past several years, Software Defined Networking (SDN) has emerged as a new and promising paradigm for the management of computer networks. While we have seen many use-cases and deployments of SDN in data center networks, wide-area networks still heavily rely on legacy routing and traffic engineering technologies. Rapidly increasing traffic demands (mainly due to increasing usage of video streaming and voice over LTE deployments), however, motivate the development of novel routing and more efficient traffic engineering mechanisms. New approaches leveraging an SDN paradigm in wide-area networks promise to mitigate many of today's limitations, inefficiencies, and scalability issues. In this paper, we give an overview of the current state of the art in Software Defined wide-area networking research and technologies and give directions and discuss ideas for future work.**

I. Introduction

Software Defined Networking (SDN) continues to attract both industry and research communities as a modern paradigm for the management and operation of computer networks.

Large-scale computer networks typically consist of a variety of devices that perform tasks ranging from switching and routing to advanced network functions, such as packet filtering or Network Address Translation (NAT). These boxes come from different vendors, are incrementally deployed, have distinctive management interfaces, and are usually configured on a per-device basis. Operating a network in this manner hinders innovation and does not only make the operation unnecessarily complicated, but also comes with high operational expenses (OPEX) [3], [4].

While there are centralized management systems that can configure systems of different types, these solutions are often vendor-specific. Software Defined Networking promises to solve many of these management headaches. It is vastly changing how computer networks are managed by providing a centralized vantage point and management interface to the entire network. In this way, network operators can configure the network from a central location using a standardized application programming interface (API) across devices and vendors.

This is mainly enabled by SDN's two main defining characteristics. First, SDN separates the network's control plane (which decides where and how a packet is forwarded) from the data plane (which then handles the packet at the device according to rules defined in the control plane). Furthermore, SDN consolidates the control planes of different devices providing a unified, centralized view of the network.

In this architecture, the centralized controller runs programs that control data plane elements using a well-defined API. This API provides an abstraction of the data plane state, as well as possible operations on this state (independently from the exact type of network device or its manufacturer).

This separation between the control and data plane allows for data plane devices to be designed in a generic and simplified way. In other words, network devices can become *simple*, *unintelligent* boxes that have the sole responsibility of handling packets based on rules as dictated by the control plane. SDN-enabled forwarding elements commonly expose a match-action abstraction. This means that network packets passing through such a device are assigned to an action, such as *forward* or *drop*, by matching a subset of the packet header to a header space defined in a rule.

While most research in SDN applications has been focused on data center settings, recently there has been an increased interest in SDN for wide-area network (WAN) scenarios. WANs are typically expensive and difficult to manage. They consist of specialized and expensive high-performance routers increasing capital expenditure (CAPEX), as well as complex configuration requiring extensive management efforts increasing OPEX and failure probability. Additionally, WANs almost always rely on a failure-prone underlying (often optical) transport network. In order to mask such router or link failure and deal with demand spikes, WANs are often over-provisioned [9]. Centralized control and load planning through SDN can help distributing load more wisely, adapting to varying demand more dynamically, and coping with failure more quickly. This can help reduce the degree of required over-provisioning and overall complexity dramatically.

In this paper, we provide an overview of the state of research in the field of Software Defined Networking for wide-area networks. We first give an introduction to the evolution of programmable networks and SDN, then present current research in the field of Software Defined WANs (SD-WAN), before concluding with directions for future research in this area.

II. History of Software Defined Networking

Underlying concepts of Software Defined Networking and programmable networks date back over 20 years. This is long before the term SDN was coined and became widespread. In particular, the programmability aspect of networks was already a *hot* research topic in the mid-1990s. We now give a brief overview of the three main steps that eventually led to the SDN technology and architecture that we have today.

A. Active Networks

With the rapid growth of the Internet beginning in the last decade of the 20th century, there was a need for new, advanced protocols as the primary use case of the Internet moved away from email and file transfer for the academic community to more generic and consumer-oriented use cases. This was mainly driven by the rise of the World Wide Web; however, the slow standardization process through the Internet Engineering Task Force (IETF) was a source of frustration for researchers. As a result, researchers believed a *programmable* network would accelerate innovation like it has been the case with high-level, general-purpose programming languages for computers. In this context, the Active Network architecture [17] was the first effort toward a fully programmable network. The idea was to inject customized programs into network nodes that are executed on the respective node when a packet traversed it. These programs were either installed on the network device and invoked by the presence and setting of a certain header field or integrated into the packet itself. This allowed third parties to develop and deploy custom functionality within the network enabling faster network innovation. While Active Networks certainly enabled network programmability, the architecture never became widely deployed. We believe that the Active Networking approach was too radical at the time, as it broke many conventional primitives widely used in networked systems (such as the concept of a protocol stack). Still, Active Networking was a very important step towards generalized network programmability. Later trends and technologies that eventually formed SDN, resemble many of the ideas of Active Networks in a less radical fashion.

B. Control- and Data Plane Separation

With the Internet rapidly growing even further in the beginning of the 2000s, Internet Service Providers (ISPs) and other network operators were looking for new ways to manage their in size and complexity increasing networks. In particular, two issues needed to be addressed:

Scalability. The Internet's inter-AS routing protocol is the Border Gateway Protocol (BGP). While AS Border Routers (ASBR) maintain a TCP session with an adjacent ASBR inside the other AS the network is peering with, all routers inside an AS must be connected in a full mesh using the AS-internal version of BGP (iBGP) to propagate routes learned at the border routers into the network. Clearly, the required full mesh leads to an undesirable quadratic scaling behavior of iBGP sessions as a function of the number of routers within the AS. Route Reflectors mitigate this problem up to some degree but are prone to route oscillations and forwarding loops. This problem led to the first systems advocating a full separation of the control and data planes; the Routing Control Platform (RCP) [2] is such a system. RCP is a centralized platform mimicking a full mesh of iBGP sessions using a central peering platform. It performs route selection on behalf of routers and communicates a single best (selected) route using standard iBGP sessions to the routers. The main advantage is that RCP provides the intrinsic correctness of a full mesh of iBGP sessions together with the vastly improved scalability of a route reflector setup.

Manageability. With higher demands in network predictability, reliability, and performance, better management abstractions and more fine-grained control over the actual path a packet takes through the network (*i.e.,* traffic engineering) became a desired functionality. As computer networks are large, complex, distributed systems with countless dependencies between modules and configurations, small, local events and mistakes can have significant, cascading impact across the entire network resulting in service deterioration, security vulnerabilities, or even complete outages [3], [4]. In order to mitigate these issues, in [4], Greenberg proposes a complete refactoring of network functionality based on configuration of network-level objectives using a network-wide view from a central vantage point with direct control over network devices resulting in a separation between the control and data planes. The main argument for this approach is that networking logic should be separated from distributed systems issues that commonly lead to failures. Casado et al. [3] further elaborates on this concept and presents a deployment of such an architecture with *Ethane*, the first instantiation of what eventually became SDN. Ethane is a management framework for enterprise networks using a single network-wide, fine-grained security policy that is centrally configured and then enforced throughout the network.

C. Control Protocols

With the emergence of various architectures separating the network control and data planes, the need for more generalized control protocols became apparent. In order to enable real-world deployment, a control protocol for hardware vendors to support was required. OpenFlow [15] was the first (and is still the most prominent) effort to balance the idea of a programmable network and some pragmatism (*i.e.,* standardization) that would enable deployment on actual networking devices. OpenFlow brought two major intellectual insights to the world of Software Defined Networking. First, it aimed at generalizing network devices and functions by a simple packet based match-action pattern. Secondly, OpenFlow introduced the notion of a Network Operating System. A software that exposes a simplified API and programming abstractions (as compared to the plain OpenFlow wire protocol) to applications that desire to control the network. Such applications can now be written in any language that the network operating system supports or use some sort of standardized API that the network operating system exposes.

As mentioned earlier, OpenFlow is not the only protocol that can be used for communication between the control and data planes. For example, in carrier networks, a Path Computation Element (PCE) architecture is often used to obtain a global view at the network topology and to centrally control traffic engineering. The Path Computation Element Protocol (PCEP) is then used in a similar fashion to OpenFlow to install forwarding state on routers. Other examples for such

communication protocols are NETCONF, RESTCONF, and the Open vSwitch Database Protocol (OVSDB).

III. Challenges in Wide-Area SDN

Wide-area networks pose special challenges to designers of SDN systems. This is mainly due to the reason that link failure and disrupted connectivity between the control and data planes in a WAN are somewhat more common than in a data center setting. Data centers often have dedicated control networks, encompass a high degree of parallel links, are scarcely exposed to physical threats (such as cable cuts), and are deployed in a safe and controlled environment. Thus, imposing strict failure resiliency requirements on SD-WANs (as opposed to data center networks) makes the use of a distributed control plane almost inevitable. Furthermore, the varying propagation delays through a WAN are typically in great excess of those in a data center. This is prolonging the time to reach a consistent updated network configuration.

We now highlight three main challenges we identified that need to be tackled when designing a SD-WAN system. Failure resiliency and scale-out behavior can be achieved with a decentralized controller architecture, however the question remains how to keep the control plane logically centralized. That is how to maintain SDN's global network view requiring consistent, centralized state among all controller instances? Furthermore, when using multiple controllers, where should these controllers be placed in regard to the different data plane elements and how many controllers are necessary at all? Finally, in the presence of high latencies between the control and data planes, how can the data plane configuration be updated such that every packet traversing the network is handled based on a single, consistent policy across all forwarding devices?

A. Distributing SDN Controller State

The separated control and data planes together with a control program and a network operating system define the overall architecture that most SDN deployments today follow. However, in this simplified picture, the centralized controller does not only often come with poor scaling behavior as it represents a single choke point, it also is a single point of failure. Control platforms with undesirable properties coming with a centralized architecture include NOX, Floodlight, and Ryu. In order to overcome these two issues, distributed controllers have been proposed [1], [12]. These controllers provide scale-out performance and fault tolerance. While in a data center setting, an SDN controller mainly has high-throughput requirements (*i.e.,* operations per second), we argue that distribution capabilities are more important for WAN SDN controllers.

Nevertheless, distributing controller instances across the network naturally also requires distributing the controller state (*i.e.,* configured match/action rules, topology information, and statistics) posing the challenge that this state must be kept consistent among instances. Furthermore, controller failure must be handled gracefully without disrupting network operation or causing state corruption. ONOS (Open Network Operating System) is a controller architecture complying with these requirements. It offers two main properties: (1) a global network view and abstraction sharing network state among all ONOS instances, (2) extensive scale-out capabilities for both performance and fault-tolerance [1]. In a network controlled by ONOS, each switch has a primary controller that programs its forwarding plane; however, the switches' state is shared among all controller instances and persisted in the distributed, eventually consistent *Apache Cassandra* key-value store making ONOS's network view eventually consistent as well. In the case of a switch or link failure, a consensus protocol provided by *Apache Zookeeper* automatically selects a new primary controller.

B. Placing Controller Instances

As alluded in the previous section, placing multiple controllers in a WAN deployment, can greatly benefit control plane latency and fault tolerance. Still, the question is how many controllers to deploy and where exactly to place them. Heller et al. [7] provide a first definition the controller placement problem and quantify the impact (particularly in terms of latency) of different placement options and strategies on real WAN topologies. The authors reduce the placement problem to three underlying, fundamental problems based on the desired metric: (1) when optimizing for average-case latency, the problem reduces to the minimum k-median problem, (2) when optimizing for worst-case latency, it reduces to the minimum k-center problem, (3) when nodes must not exceed a specific latency bound to the next controller, the problem can be stated as the maximum cover problem. While all of these problems are computationally intractable ($\mathcal{NP}$-hard), the authors show that for medium-sized networks, approximations can vastly improve latency metrics over random placement (1.4 to 1.7 times higher latency). Surprisingly, the paper also shows that in the majority of cases for the (real) wide-area networks studied, a single controller is sufficient to provide protection and restoration guarantees comparable to optical $1:1$ protection.

C. Updating SDN Switches in a Consistent Manner

While Software Defined Networks enable dynamic and customized data plane configurations, updating the data plane in a large network (*i.e.,* the packet handling rules on the switches) can cause severe inconsistencies in how packets are handled among devices. This is mainly due to the infeasibility to update the entire network atomically while maintaining full network operation. As a result, forwarding devices are typically updated in steps. Even if the initial and final configuration states are correct, network devices may be in inconsistent states during the update process. This can lead to major instabilities such as interrupted connectivity, forwarding loops, or access control violations. Moreover, intermediate update states can violate bandwidth constraints on a given path potentially causing major congestion. This happens when a link is oversubscribed between the initial and final update steps. Several solutions

have been proposed to cope with such inconsistencies. Most of them, however, rely on planning an exact update order statically before actually rolling out updates to the switches in this pre-determined order [8], [11], [14]. These solutions do not adapt to runtime variabilities within the control plane (such as varying CPU load or varying RPC delays), which can still cause inconsistencies while rolling out an update.

Reitblatt et. al. [16] address these challenges by providing a theoretical foundation and proposing a novel abstraction for network updates that guarantees consistent handling on a per-packet basis. The authors introduce a per-packet consistency model where each packet flowing through a network is guaranteed to be processed according to a single configuration. Additionally, a per-flow consistency abstraction model, where each packet belonging to a flow (*e.g.,* a TCP connection) will be processed consistently across the network, is presented.

Jin et. al. [10] propose a runtime scheduling system that uses heuristics to pick a path through a dependency graph containing the required update steps. Using this approach, a scheduler can select correct paths through this graph and the ordering of updates on the fly based on network behavior. The system predetermines valid orderings of updates and then heuristically applies them based on realtime behavior of the switches and network. Dionysus takes a two-step approach where first a dependency graph of the steps to reach a consistent final state is computed. Then, at runtime, a scheduler selects a path through this graph that becomes the order of updates maintaining consistency and correctness in the network. Dionysus significantly reduces overall update times by 53% to 88% depending on the type of update and topology compared to other solutions. Dionysus also reduces link oversubscription and thus congestion during updates significantly (41% less than comparable approaches like [8]).

IV. Benefits of Wide-Area SDN

Before continuing our discussion, we present two SD-WAN deployments from major cloud providers to illustrate the benefits that this technology offers for network efficiency and resiliency. Cloud providers like Apple, Microsoft, and Google run their services from data centers distributed around the planet. While those data centers primarily serve content to customers, there is an increasing demand for inter-data center traffic exchange. This is mainly due to the specialized and distributed character of services and subsystems requiring communication among each other, as well as extensive backup, synchronization and migration tasks across physical locations. Thus, inter-data center (wide-area) networks are a critical infrastructure for such providers and are commonly over-provisioned and consequently underutilized by around 30% to 60% in order to cope with failure [8]. The global network view and fine-grained control over routing decisions that SDN promises can be leveraged to achieve centralized traffic engineering optimizing for more efficient use of the expensive network resources. Microsoft and Google both employ SDN technologies for their inter-data center networks and have published technical details of their systems.

The first system, Software-driven WAN (SWAN) [8], from Microsoft, is a centralized traffic engineering (TE) solution that uses fine-grained policy rules and a centralized scheduler to carry significantly more high-priority traffic (*e.g.,* traffic for interactive applications) while maintaining fairness among services of the same class. SWAN addresses shortcomings of today's primarily used decentralized TE practice (*i.e.,* MPLS-TE in conjunction with ECMP routing) and makes a strong point for using a global network view and SDN in order to solve the global and network-wide problem of TE. SDN's global network view is used here to find globally optimized bandwidth to path assignments through the network. Fine-grained control in SDN is used to make and enforce bandwidth reservations on a meticulous, per-application basis. By doing so, the need for over-subscription is drastically reduced, and resources are used more effectively. As a result, SWAN can carry about 98% of the theoretically possible network traffic while MPLS-based technologies carry only around 60%. Furthermore, the authors identify the need for frequent data plane updates in order to maintain utilization at a high level and address this problem by presenting a consistent update algorithm.

The second deployment, B4, is Google's globally-deployed software defined inter-DC WAN solution [9]. The system exploits three unique properties of their network to deploy a centralized TE solution achieving high link utilization and a two times to three times efficiency improvement relative to standard (decentralized MPLS-based) practice. First, Google controls all IP-layer network equipment all the way to the edge of the network. Second, most bandwidth is used for low-priority, planned, large replication tasks between sites. Finally, the network only consists of a few dozen sites making a centralized control-plane approach feasible. B4 employs custom switches controlled by a slightly modified version of OpenFlow. The network controller dynamically reallocates bandwidth for shifting application demands and also provides dynamic rerouting in the case of link or switch failures. The SDN-based control plane also integrates with standard legacy routing protocols that make it possible to completely disable centralized TE while keeping the network fully operational. Many B4 links run at 100% utilization and all links average at 70% utilization over extended time periods.

While, both B4 and SWAN use SDN in the context of inter-DC wide-area networks and present similar high-level architectures focused on traffic engineering for optimized resource usage of expensive wide-area links, their secondary focuses are vastly different. Hong et al. with SWAN make significant contributions in the field of update consistency (cp. section III-C) and leverage frequent data plane updates together with correctness guarantees to maintain high utilization over extended periods of time. The authors of B4 put an emphasis on integrating a (TE-centered) SDN control plane with existing legacy routing protocols (in this case BGP and IS-IS) as a fallback mechanism. The Routing Application Proxy (RAP) of B4 interfaces with standard Quagga software switches and receives BGP and IS-IS control messages directly

from the SDN-enabled switches through the software slow-path.

V. Expanding Beyond A Single Domain

Intra-domain SDN solutions mainly tackle problems around shortcomings of legacy interior gateway routing protocols or traffic engineering practices (such as MPLS-TE) to make better use of expensive wide-area links (as explained in section IV). The problems addressed with inter-domain deployments are somewhat different. Most of these problems stem from the de-facto standard inter-domain routing protocol BGP, in particular from its exterior version eBGP. BGP, as the standard protocol for almost all inter-domain peering in the Internet, keeps the Internet running despite extremely complex operational and financial interests from the various network operators. This means that BGP does not only come with technical challenges such as intra-domain scalability issues [2], slow convergence times, route flap damping, or route oscillation; it also imposes operational challenges for network operators because they have to use BGP's embedded mechanisms like local preference attributes, Multi-Exit Discriminators (MED), or AS-PATH prepending to implement complex policies representing their business objectives. Those instruments need to be manually configured and are error-prone, which commonly leads to (sometimes global) routing instability or divergence.

Similar to the intra-domain case, SDN can offer new opportunities to overcome these problems. Again, a global view of the network can be leveraged to make network-wide decisions over how packets are forwarded. Since autonomous systems are operated by different parties with different network equipment, policies, and business objectives that cannot be shared with other networks, applying logically centralized control across multiple administrative domains is somewhat more complex and poses different challenges compared to the intra-domain case.

One of the first papers on this issue was published by Kotronis et al. [13]. The authors propose a rather drastic approach that is to completely outsource inter-domain routing logic to a third party (*i.e.,* a trusted contractor). This contractor can specialize in the practices of Internet routing, freeing network operators from complex BGP-related administration tasks to comply with its policies. This approach has three main advantages: (1) as the contractor performs routing control for multiple ASes, it can optimize global routing using its birds-eye view, (2) it can detect policy conflicts and help with collaborative troubleshooting between ASes, (3) as inter-domain routing would be centrally controlled, existing mechanisms (*i.e.,* eBGP) could be improved among networks having the same contractor without disrupting connectivity, thus accelerating innovation in inter-domain routing.

A more recently presented system uses some of these ideas but applies them in a more natural way that keeps network operators entirely in control of their peering policies. Their approach is applied at Internet Exchange Points (IXP). An IXP is a physical infrastructure where network operators exchange traffic and BGP route information between their networks (ASes) over a common layer 2 fabric to which all networks are connected. IXPs present a natural starting point for wide-area, inter-domain SDN deployments because SDN deployment at a single IXP immediately affects a large number of providers.

SDX (Software Defined Internet Exchange Point) [6] is an IXP that consists of an SDN controller instead of a route server and an SDN-enabled switch instead of the plain L2 fabric. Each AS connected to this SDN switch can now write its custom policies that are being enforced on its private virtual switch. The SDX controller composes these rules (which are expressed in the Pyretic programming language) behind the scenes and installs the actual forwarding rules on the SDN-enabled IXP switch. Furthermore, SDX integrates with legacy BGP routing through its integrated route server. This also means that not every ISP at an SDX must use the SDN capabilities of the SDX system but can also exchange its routes using standard BGP, like in a traditional IXP. The SDX controller runs integrity checks on all requested policies to make sure that they do not create conflicts or forward traffic to prefixes that have not been advertised by the receiving AS. SDX however, explicitly enables overwriting BGP default routes for use cases like application-specific peering, which can be realized by not only using the destination IP prefix for route selection, but also other match fields that are available in OpenFlow. The policy composition of SDX can generate significant numbers of rules to be installed that are well beyond the capabilities of modern OpenFlow-enabled switches (in the order of tens of millions). The authors further improved their system to an industrial-scale SDX (iSDX) [5] that leverages various rule transformations, outsourcing of rules to ASBRs, and recent advances in switch hardware (L2 bit mask matching) to encode reachability information in a packet's L2 header further reducing required centralized data plane state.

VI. Research Challenges and the Future of SD-WAN

After giving an overview of the history of SDN and presenting the state of research in SD-WAN, we identified areas with ongoing challenges in the context of wide-area networking that we believe can greatly benefit from an SD-WAN approach. While most of these challenges have been studied in the context of a traditional networking paradigm, we believe leveraging the main tenets of SDN can have significant impact on the way we handle common issues in wide-area networks. We now briefly present some of these potential research areas:

Distributed Control Planes. While [7] discusses the placement problem and [1] presents a fully distributed controller architecture, it remains unexplored how distributed control planes behave in real-world wide-area topologies that are significantly harder to study compared to standard data center topologies such as CLOS or FatTree.

Data Plane Load Balancing. Wide-area networks heavily use parallel links based on constraints in the underlying transport network (*e.g.,* wavelength availability). SDN's global view of the network and more fine-grained forwarding capabilities

offer the opportunity for better abstractions and load balancing schemes in comparison to ECMP routing.

Traffic Engineering and Network Monitoring. The SDN control plane has a global, centralized view of the network and thus also can access network-wide statistics and properties. With centralized traffic engineering solutions, this data can be leveraged to find globally optimal path assignments. Traditional traffic engineering mechanisms such as RSVP-TE or LDP for MPLS rely on the local, limited view of the network from the ingress router. We must find ways to use such network-wide properties with application-specific traffic engineering solutions effectively.

Data Plane Fault Tolerance and Low-Latency Routing. Programmability in SDN offers possibilities to implement custom, fast, and efficient adaptive routing schemes that optimize for low end-to-end latency, as well as failure recovery mechanisms that can be achieved with low communication overhead and little controller interaction. Nevertheless, it remains unsolved how to pick better paths in real time in the presence of link failure or congestion, which are scenarios that are common in wide-area networks.

Packet-Optical Convergence. Multilayer coordination between the transport and IP/MPLS packet layers of a WAN is a promising approach towards more optimized and easier to manage carrier networks. Programmability in software defined networks lowers the burden for operators to integrate the network layers. Reachability, performance, and protection properties from both planes can then be used to optimize traffic engineering and fault tolerance. However, integrating these two vastly different systems with their custom control software and very different experience and expertise of their operators poses various challenges.

Internet-Scale Attacks. Large-scale, Distributed Denial of Service (DDoS), attacks have become widespread in today's Internet and are one of the major threats for network and service providers as they may cause severe outages coming with significant business consequences. A global network vantage point and centralized data collection (especially in inter-domain scenarios) can be valuable to identify optimal locations to monitor and block large-scale attacks much closer to the attacker (as opposed to the target) than normally possible.

VII. Conclusion

In this paper we gave an overview of the current state of research in the field of Software Defined Wide-Area Networking along with a discussion of potential future research topics. We believe, that, especially with growing industry attention due to ever-growing traffic demands, SD-WAN will remain an area of high interest over the upcoming years.

While we are still in the beginning phases of research in this area, we see considerable space for improvement of wide-area network performance, reliability, and scalability through Software Defined Networking technologies. In particular, augmenting visibility in WANs through monitoring and telemetry technologies as well as the integration of the control planes of the packet and optical (transport) layers of WANs may have significant impact on the future operation of wide-area networks.

As WANs typically consist of specialized, expensive, high-performance hardware and are built incrementally, we, however, project that the adoption of radical new SDN technologies and deployments happen at longer time scales than we have seen it in data center networks, where completely new deployments are needed more often. Although, through technologies like PCEP and OpenFlow, a lot of equipment is already SDN-enabled potentially shortening adoption times.

References

[1] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. ONOS: Towards an Open, Distributed SDN OS. In *Proc. of the Third Workshop on Hot Topics in Software Defined Networking*, 2014.

[2] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and Implementation of a Routing Control Platform. In *Proc. of the 2nd Conference on Symposium on Networked Systems Design & Implementation*. USENIX Association, 2005.

[3] M. Casado, M. Freedman, and J. Pettit. Ethane: Taking control of the enterprise. *ACM SIGCOMM CCR*, 37(4), 2007.

[4] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *ACM SIGCOMM CCR*, 35, 2005.

[5] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever. An Industrial-Scale Software Defined Internet Exchange Point. In *13th USENIX Symposium on Networked Systems Design and Implementation*, mar 2016.

[6] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. SDX: A Software Defined Internet Exchange. In *Proc. of the ACM SIGCOMM 2014 Conference*, 2014.

[7] B. Heller, R. Sherwood, and N. McKeown. The Controller Placement Problem. In *Proc. of the 1st Workshop on Hot Topics in Software Defined Networks*, 2012.

[8] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving High Utilization with Software-driven WAN. In *Proc. of the ACM SIGCOMM 2013 Conference*, 2013.

[9] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-deployed Software Defined Wan. In *Proc. of the ACM SIGCOMM 2013 Conference*, 2013.

[10] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer. Dynamic Scheduling of Network Updates. In *Proc. of the ACM SIGCOMM 2014 Conference*, 2014.

[11] N. P. Katta, J. Rexford, and D. Walker. Incremental consistent updates. In *Proc. of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13. ACM, 2013.

[12] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. In *Proc. of the 9th USENIX Conference on Operating Systems Design and Implementation*. USENIX Association, 2010.

[13] V. Kotronis, X. Dimitropoulos, and B. Ager. Outsourcing the Routing Control Logic: Better Internet Routing Based on SDN Principles. In *Proc. of the 11th ACM Workshop on Hot Topics in Networks*, 2012.

[14] H. H. Liu, X. Wu, M. Zhang, L. Yuan, R. Wattenhofer, and D. Maltz. zupdate: Updating data center networks with zero loss. In *Proc. of the ACM SIGCOMM 2013 Conference*, SIGCOMM '13. ACM, 2013.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM CCR*, 38(2), mar 2008.

[16] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. Abstractions for Network Update. In *Proc. of the ACM SIGCOMM 2012 Conference*, 2012.

[17] D. L. Tennenhouse and D. J. Wetherall. Towards an Active Network Architecture. *ACM SIGCOMM CCR*, 37(5), oct 2007.