Middleboxes, and Various Tools for Robust Deployment of Software Defined Network (SDN)

Nishtha
Department of Computer Science,
Rajiv Gandhi Govt. Degree College, Chaura Maidan, Shimla, India

Abstract: The segregation of control and data planes in Software Defined Network (SDN) has led this network architecture to leverage numerous benefits from easy innovations, high scalability, simplified network management, and robust fault tolerance are some of the numerous advantages provided. This emerging architecture has consequently resulted in reconsidering our viewpoints regarding network architecture and design. In this context, this paper discusses and lists various middleboxes, switch platforms, emulation, and simulation, code verification, debugging tools, languages, used, and common APIs used for Software Defined Network (SDN) deployment.

IndexTerms **– Middleboxes (MBs), Switch Platforms, Emulation Tools, Simulation Tools**

1. INTRODUCTION

The partition of control and data plane in Software Defined Network (SDN) architecture has resulted in governing this architecture through software-based logically centralized control known as the controller. In this architecture, a protocol, OpenFlow developed by ONF (Open Networking Foundation) or any other proprietary protocol is utilized that authorizes the controller to install rules in the forwarding element and so the network traffic propagates [1]-[3].

For leveraging the SDN-based services various tools and middleboxes are required. The prime objective of this paper is to discuss middleboxes and various tools such as switch platforms, emulation, and simulation, code verification, debugging tools, languages, and used and common APIs available for establishing robust SDN services.

Various sections in this paper are separated as section 2 discusses middleboxes, and various development tools required for leveraging SDN services are discussed in subsequent sections under section 3. Section 3.1 discusses switch platforms, emulation and simulation tools in section 3.2, code verification, debugging, and updation tools in section 3.3, and languages, used, and common APIs are discussed in section 3.4 followed by a conclusion in the end.

2. MIDDLEBOXES (MBs)

Middleboxes are data plane devices that reside between traffic flow paths and are used to provide additional packet processing services such as compression, WAN optimizers, virus scanning, proxies, intrusion detection, prevention, encryption systems, network and application-level firewalls, and application-specific gateways [4]-[8]. Middleboxes are vertically integrated which means that the software required is closed-source and the hardware required is proprietary [2], making the network inflexible. It raises many issues regarding their scaling, placement, etc. [5].

The existing trend is towards eliminating middleboxes and embedding their functionality directly into the network controller. Using a web Ad-insertion application, a control framework is demonstrated that provides network services in an error-free and efficient manner. It automatically processes modules and networking devices and at the same time manages various performance requirements of network services [4].

Software centric approach to middlebox architecture is proposed where several MBs share the same hardware, managed by a logically centralized management that provides an open and extensible API [5].

APLOMB is a service that allows outsourcing of enterprise middlebox processing to the cloud. With third-party involvement, enterprises are relieved from infrastructure-related problems concerning cost, management, etc. [6].

Deploying middleboxes at choke points raises issues related to robustness, correctness, and efficiency. The MB architecture follows a specific model in which MBs are deployed as waypoint services and OpenFlow switches at choke points. So, multiple routers may access the services that are implemented at waypoints from anywhere in the network by redirecting traffic for additional processing to those services through action API [7].

Current ad hoc mechanisms for MB control are unsuitable. An alternative, software-defined MB networking framework that supports MB scaling and live network migration has been proposed [8].

OpenFlow provides a fixed and narrow data plane. MBs may be an alternative to the data plane & may supplement the data plane but, MBs lack a match and action interface in order to interact with the control plane. Slick [9] is a network programming architecture that by providing an interface for communication between the two also separates processing at the controller and at middleboxes. MBs are not vertically integrated, and new software may be loaded dynamically onto them.

Network-wide policies may not be suitably applied in networks in which middleboxes are dynamically modified depending on traffic state. FlowTags is an SDN-based architecture in which middleboxes add Tags to outgoing packets that are used on switches and other middleboxes for stable policy execution in the dynamic environment [10].

Network Functions Virtualization (NFV) using virtualization technology aims at the implementation of network functions to industry-standard using high-volume servers, switches & storage. NFV with SDN, using virtualization and cloud computing

technologies, unify the functions that provide benefits such as cost reduction and enhanced service delivery, etc. As a result, new and innovative services may be provided [11]-[12].

III. SDN DEVELOPMENT TOOLS

SDN is a new architecture, and various tools are being developed for SDN. Various switch platforms, emulation, and simulation, code verification, and debugging tools available in SDN are discussed in this section. SDN supports a programming environment that allows the creation of networks dynamically. In this section, commonly available tools for developing SDN-based services and their associated protocols are also discussed.

3.1 Switch Platform

SDN switches are developed by different vendors that provide services over SDN. A multilayer software switch, Open vSwitch is a switch platform with a standard management interface and open forwarding functions that is suitable to be used aa s virtual switch in a VM environment [13].
Table 1 provides a list of most of the currently available vendor switches and their specific characteristics:

Table 1. Common OpenFlow Switches by different vendors

.N.	Vendor	Switch Model	Specific Characteristics
1.	Pica8 [14]	P-3297, P-3922, P-3930, P-5101, P-5401 1GbE / 10 Gb/ 40 GbE Open Switches	- Used in the cloud or virtualized data centers. - Easy integration with traditional networks as well as SDN can be done. - Open vSwitch (OVS) runs as a process within PicOS, Pica8 operating system.
2.	Hewlett-Packard [15]	HP Flex Fabric 12900 switch series, Flex Fabric 11900 Switch Series, Flex Fabric 5700 Switch Series, 8200zl, 5400zl, and 3500yl	- Provides scalability & performance for next-generation modular DC. - Provides low latency for campus networks. - Provides high security, resiliency & multiservice support for large campus & branch offices. - These switch series have a common platform architecture. - Includes networking features such as QoS and security. - Suitable for applications throughout the network topology.
3.	IBM [16]	IBMFlux System series RackSwitch G8264	- Provides high scalability. - Used for HPC, and other applications requiring high bandwidth and low latency.
4.	Brocade [17]	NetIron CES 2000 Series	- Best suited for Carrier Ethernet service delivery at the network edge.
5.	NEC [18]	PF5240, PF5248, PF5820	- Provides immense network virtualization, security, and programmable capabilities.
6.	Juniper [19]	Junos EX-Series Three EX9200 switches are: EX9204, EX9208, EX9214	- Best use in cloud applications, server virtualization, and in campus and data center core and aggregation networks. - Provides simple and secure access on campus and by simplifying operations it can easily accommodate fast-changing business requirements in data centers. - EX9208 is one of the most acknowledged Ethernet switches used for Technology Innovations.
7.	Pronto [20]	3290, 3295,3920 and 3780	- Supports dual mode of operation layer 2/layer 3 and OpenFlow and thereby enabling gradual integration of SDN architecture in an existing network. - Reduced costs.
8.	Arista Networks [21]	Arista 7150 Series Switches	- Best use in data center networks, and HPC applications and provides low latency and high performance. - First network–wide virtualization platform.
9	Extreme Networks [22]	BlackDiamond 8000 Series	- Provides a two-tier network that reduces management overhead, operational complexity, and capital expenditures.
10.	ADVA Optical Networking [23]	FSP 150	- ADVA's FSP 150 Network Interface Devices allow the wholesaling of Ethernet services by providing network separation with in-service monitoring and alarming, including timing distribution.
11.	Net gear [24]	GSM7352Sv2	- Combines resiliency and security for enterprise security and also provides nonblocking performance. - In the event of a power supply failure, the switch can immediately shift to an external RPS device while the internal power module is replaced for 100% uptime.
12.	NoviFlow Inc 25]	Novikit 100 NoviKit 200, NoviVswitch 1132, 1248	- First High-Performance switch that supports OpenFlow v1.3. - Used for commercial purposes including cloud computing, SDN, and Network Function Virtualization (NFV).
13.	Cyan [26]	Z-Series Packet-Optical Transport Platform	- Delivers orchestration, visualization, and scale to networks that are static and hardware-driven. - Used in carriers, enterprises, governments, and data centers network. - Cyan's open platforms provide multi-vendor control and visibility to network operators, making service delivery more efficient and profitable.
14.	Centec Networks [27]	V330 series, v 350 series Switch	- The V330 adopts industry-standard Open vSwitch (OVS) as the OpenFlow switch software stack with a standard OpenFlow compliant interface to OpenFlow controllers. - Deployed in Data Centers to offer cloud computing services.
15.	Transmode 28]	TM-Series	- Used in a Packet-optical network supporting Layers 0. - Also provides Enlighten multi-layer management suite to manage TM-Series networks. - Transmode's SDN- enabled packet-optical networks to support fully automated multi-layer Path Computation Element (PCE) based path provisioning for all MPLS-TP, Ether,net, and Layer 1 services.

Hierarchical switch architecture that enables the development of a large logical switch from smaller OpenFlow switches has been proposed. This may increase the path capacity of a logical switch that otherwise comes as a limitation of the restricted size of flow tables [29]. Time and accurate measurements are the two important factors on which network traffic depends. A measurement model where packets are matched for the highest priority by the switches is proposed. A hierarchical heavy hitter algorithm that identifies large traffic aggregates and maintains the balance between measurement accuracy and switch overhead is designed [30]. A CPU when used in commodity switches is used to handle control as well as data plane traffic [31].

SDN using OpenFlow a programmable architecture has been proposed that abstracting core transport node into a programmable virtual switch solves most of the problems faced by packet-optical network service providers [32]. OpenFlow switches from three different vendors have been studied to analyze the impact of some important parameters such as CPU, TCAM type and size, and firmware on application performance [33].

To provide a global network view, a number of counters are needed for every rule in a flow table which is implemented in hardware as an ASIC. ASIC-based counters are inflexible, therefore, mayn't be changed easily. These counters may be replaced with a small buffer in case the data plane has a fast connection to a general-purpose CPU with cost-effective memory (DRAM) [34].

3.2 Emulation and Simulation Tools

Alterations in protocol, configurations, software, or hardware of a network may affect the network. Therefore, before using the proposed changes their performance is verified using a network emulation tool [35].

ModelNet [36] is a large-scale emulation system that allows researchers to deploy unmodified software prototypes and verify them for Internet-like conditions. User-specified Operating systems and software which run in edge nodes are configured to route their packets through a set of ModelNet core nodes. These core nodes adjust the traffic as per the bandwidth, congestion constraints, latency, and loss profile of the target network topology. Mininet is a network emulation platform that has the ability to create a virtual OpenFlow network, controllers, switches, hosts, and links on a single real or virtual network. It uses Operating System virtualization that allows large networks to be prototyped on the restricted resources of a single machine such as a laptop [37].

SDN-based applications including new programming languages, frameworks, debugging, static analysis techniques, and VM and container-based emulation tools are tested and evaluated using a simulation-based tool, fs-sdn [38].

3.3 Code Verification, Debugging, and Updation Tools

Some of the important tools for code verification, debugging and network updation are:

FlowChecker [39] is a method for sensing bugs related to communication in Message Passing Interface (MPI) libraries.

OFRewind [40] is a system capable of recording events that can be retrieved back so that the events causing network errors could be checked. OFRewind helps operators to reduce software errors, identify data-path limitations, and even locate configuration errors.

To check a class of bugs that causes errant packets to be processed in-network, the debugger tool Ndb has been implemented which uses breakpoints and packet backtracking. It changes the state of an errant packet and by live observations may even detect the main cause of the error. OFRewind eliminates errors by retrieving the events whereas, Ndb eliminates by live observation [41].

Header Space Analysis [42] is a framework that automatically checks errors by examining network specifications and configurations. By evaluating the data plane configurations Header Space Analysis detects connectivity as well as isolation errors. An automation tool NICE [43] combines symbolic execution and model checking to verify controller code and is used for testing applications. It combines symbolic execution and model checking to verify controller code. VeriFlow [44] is a network debugging tool that dynamically checks bugs when a forwarding rule is inserted by forming a layer between an SDN controller and network devices. The tools discussed such as Header Space Analysis and even VeriFlow are unsuitable for dynamic verification of network configuration in large networks. A machine-verified SDN controller has been developed with which the programmers by using the NetCore programming language can specify the behavior of the network thereby providing programmers with static guarantees [45].

Network configuration updates are regularly required in a network to move programs to a new controller platform, eliminate bugs, or address performance issues. Abstracting network updation in SDN makes programs reliable, easy to design, and may be validated using automated tools. Several useful update operations are proposed in [46]. An approach that uses high-level interfaces and abstraction allows programmers to implement consistent updates and reduces the overheads. Thus, these updates are cheap and faster [46].

TIMECONF is an approach that uses time as the main factor for performing network updates [47].

The existing approach followed to update a controller is unsuitable and may cause packet loss, increase latency, and even compromise correctness. HOTSWAP [48] is a system that continuously upgrades SDN controllers by acting as a hypervisor that sits between the switches and the controller. It maintains a log of network events in order to upgrade the new controller from the old one. By bootstrapping the new controller, it examines which parts of the network state may be reused in the new controller.

During the consistent update, the packet entering the network should either obey the old or the new policy but, not a mixture of two policies. An algorithm for incremental consistent network update has been presented which maintains a balance between update time and rule-space overhead. By dividing a global policy into a set of slices and increasing the number of slices, the rule-space overhead is substantially reduced [49]. A protocol that sequence-wise updates rules in each switch have been presented [50].

3.4 Languages / Tools for API development

SDN being programmable networks are controlled by software components known as the controller. Different controllers are developed using different programming languages. Most of these controllers are developed using general programming languages like Java and Python. But, the languages that SDN needs be expressive as well as analyzable. Therefore, SDN-derived languages are also being developed.

A high-level domain-specific language, Flow-Based Management Language (FML), has in-built policy operators which allow/deny certain flows [51]. Some languages such as Frenetic and Nettle also help in writing better code [52], [53].

Frenetic and Network Core Programming Language (NetCore) are also complete programming languages for SDN environments that support dynamic policies and states [52], [54]. Frenetic is a language for packet forwarding that combines the features of declarative query language and functional stream-processing [52]. Procera is a high-level control language for policy declaration and may be used by network operators for dynamically implementing network control policies [55].

Flog is a language that has mixed features of FML and Frenetic [56]. The feature of logic programming has been taken from FML while the mechanisms to query network state, process queries, and generate packet-forwarding policies which are automatically installed into switches have been taken from Frentic. Using a compiler and a run-time system, NetCore language enables rules to be automatically installed to individual switches [57].

Most of the controller platforms e. g. NOX, and Beacon provide low-level, imperative, and event-driven programming interfaces. The programs which react to network events by installing/uninstalling rules to each switch individually are not preferable [41], [58].

FatTire [59] is a language in which fault-tolerant network programs may be written by using regular expressions. The developers have the benefits of specifying the paths that packets take in the network as well as the degree of fault tolerance required.

EtherPIPE [60] is a character network I/O device by uses UNIX commands and allows to access network traffic data in the same manner as a file is accessed.

A network-programming language FlowLog has the capabilities to express and analyze. It also allows the reuse of pre-existing code [61].

SDN architecture also provides Open APIs for communication between the controller and the data plane as well as the controller and the applications. Consequently, SDN APIs have been developed by commercial vendors and Table 2 provides the list of common APIs.

Table 2. List of SDN APIs by commercial vendors

S.N.	API Name	Vendor	API Details	Important characteristics
1.	EOS API Arista's newest interface, EOS API (eAPI) [62]	Arista	Arista EOS offers a rich set of programmable interfaces including: Linux shell access and APIs OpenFlow, DirectFlow Sysdb APIs Python, Perl scripting, Advanced Event Management (AEM) EOS SDK JSON-based eAPIs, CLI, SNMP, XMPP. - eAPI uses JSON as a third-party client. Therefore, language-independent.	- Provides multiple programmable interfaces for application development. - These interfaces can be used by applications running on the switch or external to the Extensible Modular operating System (EOS). - Used to access any state and configure any properties on the switch. - Provide full ability to automate and control the data center. - JSON as a third-party client and the simplicity of API makes it language-independent. - Used to access any state and configure any properties on the switch. - Will remain compatible with future versions of EOS. - Easy to be integrated into any existing resources and workflows. - EOS API (eAPI) provides full programmatic control over EOS. - Flexible and easy to use.
2.	Floodlight Northbound API [63]	Big Switch Networks	XML API A RESTful API between the controller platform and the SDN Applications	- When we run the Floodlight, the controller and the set of configured module applications will run, too. - Any application can interact (retrieve information and invoke services) with the controller by sending REST commands. Any application can interact (retrieve information and invoke services) with the controller by sending REST commands.
3.	VMware APIs 1. VMware CIM API. 2. VMware vSphere Site Recovery Manager API. 3. VMware VIX API: 4. Programming API. 5. Scripting API [64]	VMware	Java and Microsoft .NET (C#) code for web services and client applications. -VMware VIX API: also includes C, Perl, and COM bindings, Visual Basic, VBscript, C#). with code samples	1. Common Information Model (CIM) API is used for building server management applications. 2. VMware vSphere - Site Recovery Manager API. - An API for developing applications such as third-party software, after failures, can test or start recovery plans and is also able to control protection groups. 3. VMware VIX API: Can be run on different systems for automating virtual machine and guest-OS operations. 4. Programming API: A legacy interface that works with VMware Server 1.0 for automating virtual machines. - Replaced by VMware VIX API. 5. Scripting API: The legacy interface is replaced with vSphere SDK or the vSphere Web Services SDK.
4.	VellOS RESTful API [65]	Vello	JSON or XML	- Used in web communications either at the server-side or client-side. - These APIs use JSON schema as the format of message passing.
5.	Quantum API [66]	OpenStack	JSON, XML Like other OpenStack CLI tools, the 'quantum' tool is just a basic wrapper around the Quantum API, so any operation that can be performed via the CLI has an equivalent API call that can be performed programmatically.	- Quantum provides an interface for communication between devices managed by the OpenStack services. - It is a virtual network service [57]. - The OpenStack Quantum API is defined as a ReSTful HTTP service. - The API takes advantage of all aspects of the HTTP protocol (methods, URIs, media types, response codes, etc.) and the providers are free to use protocols with existing features.
6.	Junos XML API [67]	Juniper	XML	- Lunos XML API configures devices that are running Junos OS. - Uses XML concepts.
7.	iRules [68]	F5	Uses Tcl scripts	- iRules is a flexible event-driven programmable interface.

			-iRules are user-created Tool Command Language (Tcl) programs or scripts that are assigned to a Virtual Server and run (or are triggered) by one or more user-specified events related to that Virtual Server.	- These event-driven Tcl scripts contain a number of commands which may be used to make load balancing decisions, modify packet content, direct traffic flow, collect statistics, and anything from layer two to another layer above it. - Provides a great deal of power and control by improving security, and resiliency, and by increasing the number of applications in the data center.
8.	InSite SDK API 1. Extreme XOS InSite (APIs). 2. The Ridgeline InSite SDK API [69]	Extreme	-SOAP/XML. -Ridgeline InSite Software Development Kit (SDK) Provide comprehensive APIs. These interfaces are used by third-party, partner, and OSS/BSS applications to retrieve information network-wide.	1. Extreme XOS InSite API: device-to-device management communication by allowing applications the ability to interface directly with Extreme Networks switches. 2. The Ridgeline InSite SDK API: Provides application-to-management communication. The network-wide information retrieved using the APIs allows network administrators to create Service Oriented Architecture solutions by eliminating the gap between applications and business logic. - Both the APIs enable reliable and secure communication using XML messages. - InSite makes the integration of the network resources with high-level applications and business software easier.
9.	Open Automation Framework API [70]	Dell / Force 10	Smart Scripting allows development in TCL, ZCL, Expect, PERL, Python, and UNIX shell scripts.	- By using a Smart Scripting network, administrators may create scripts in a number of languages. Thus, it eliminates the need to learn proprietary scripting languages. - Allows fast development as well as deployment of custom scripts - By offering a UNIX environment smart scripting is useful for cloud administrators who interact directly with the UNIX shell. - Allows developers to include a convenient set of API function libraries. - Provides industry-based standards. - An automation technology that simplifies the management of dynamic virtual data centers. - Reduces risk and overhead.
10.	1. Open Networking Environment Platform Kit (onePK) API 2.Nexus 1000V XML API [71]		Java, C, Python Scripting OpenFlow is built on a onePK consistent manner.	1. An Open API that uses the network functions in network devices and network operating systems for communication in a consistent manner. - The infrastructure layer abstracts the platform-specific details of different NOS that are being programmed or being run by application programmers. - Improve speed. - Fast adaptability. 2. Nexus 1000V XML API is being developed to program the virtual network overlays.
11.	ADX OpenScript API [72]	Brocade	Open Script, Perl Scripts. Open Script API supports object-oriented programming.	- Open source and event-driven. - The functionality of load-balancing such as content manipulation and server selection may be included in the Open Scripts by using ADX Perl Extensions. - Perl programs are robust and do not crash on accessing values even in case of error conditions. - Both OpenScript, as well as Perl Scripts, are Event-Driven.
12.	SDN API Management [73]	Apigee	-JavaScript - It includes two components: 1. embedded API management software 2. stand-alone API platform for multi-vendor SDN network integration and analysis	- Used in Data Centers. - Northbound API that creates, manages, and is able to scale, applications for the programmable data center.

IV. CONCLUSION

This paper summarizes most of the middleboxes and tools available in the literature that helps in the robust deployment of SDN. Open interfaces for communication between the controller and the applications provided by SDN allow the development of distinct custom-built applications for network management. The modification in middleboxes and various tools available for SDN or their latest development will definitely leverage new and easy network capabilities and services in it.

REFERENCES

[1] M. Shirazipour, Y. Zhang, N. Beheshti, G. Lefebvre, and M. Tatipamula, 2012. OpenFlow and multi-Layer extensions: overview and next steps. EWSDN, 7-12.
[2] Sood, M. and Nishtha, 2014. Traditional versus Software Defined Networks: A review paper, Published in IJCEA.
[3] Open Networking Foundation (ONF): https://www.opennetworking.org.
[4] Lee J., Tourrilhes, J., Sharma P. and Banerjee, S. 2010. No more middlebox: integrate processing into network. SIGCOMM'10, ACM, 459-460.
[5] Sekar, V. and Ratnasamy, S. Reiter, M. K., Egi, N. and Shi, G. 2013.The middlebox manifesto: enabling innovation in middlebox deployment. 10th ACM Workshop on Hot Topics in Networks HotNets-X, ACM.
[6] Sherry J., Hasan, S. Scott, C, Krishnamurthy, A., Ratnasamy, S. and Sekar, V. 2012. Making middleboxes someone else's problem: network processing as a cloud service. SIGCOMM'12, ACM, 13-24.

[7] Gibb, G. Zeng, H. and McKeown N. 2011. Initial thoughts on custom network processing via waypoint services. 3rd Workshop on Infrastructures for Software/Hardware co-design.

[8] Gember, A, Prabhu, P, Ghadiyali, Z.and A. Akella, A. 2012. Toward software-defined middlebox networking. HotNets-XI, ACM, 7-12.

[9] Anwer, B. Benson, T. Feamster, N. Levin D.and Rexford J. 2013.A slick control plane for network middleboxes. 2nd ACM SIGCOMM workshop HotSDN, ACM, 147-148.

[10] OpenFlow-enabled SDN and Network Functions Virtualization: https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf

[11] Fayazbakhsh, S. K. Sekar, V. Yu, M and Mogul, J. C. 2013.HotSDN '13, ACM, 19-24.

[12] White Paper: Network FunctionsVirtualization—Everything Old Is New Again.

[13] Risdianto A. C. and. Mulyana E, Implementation and analysis of control and forwarding plane for SDN. 2012, 7th International Conference TSSA, 227 - 231.

[14] Pica 8 switches: http://www.pica8.com/open-switching/1gbe-10gbe-40gbe-open-switches.php

[15] HP switches Flex Fabric series, 8200zl, 6600, 6200zl 5400zl, and 3500/3500yl: http://h17007.www1.hp.com/IN/en/networking/solutions/technology/sdn/portfolio.aspx#.VNLX0hv9nmQ

[16] IBM Rack Switch G8264: http://www-03.ibm.com/systems/networking/switches/rack/g8264/

[17] Brocade switches: http://www.brocade.com/products/all/switches/product-details/netiron-ces-2000-series/index.page

[18] NEC switches: http://www.necam.com/sdn/doc.cfm?t=PFlowPF5240Switch

[19] Juniper EX series:http://www.juniper.net/techpubs/en_US/release-independent/junos/topics/reference/specifications/host-subsystem-ex9200.html

[20] Pronto3290, 3295, 3920, 3780: http://www.iwnetworks.com/pronto-3780.php

[21] Arista 7050 Series Switches: http://www.aristanetworks.com/en/products/7050series

[22] Extreme Networks BlackDiamond 8000 Series: https://www.extremenetworks.com/ libraries/services /BlackDiamond8800SeriesSwitchesHardwareInstallationGuide-100284Rev06.pdf

[23] ADVA Optical Networking FSP 150:http://www.advaoptical.com/~/media/Resources/Fact%20Sheets/ Carrier%20Ethernet%20Access.ashx

[24] Net gear GSM7352Sv2: http://support.netgear.com/product/GSM7352Sv2

[25] NoviFlow Inc NoviKit 200: http://www.nvc.co.jp/pdf/product/noviflow/NoviKit200Datasheet.pdf

[26] Cyan Z-Series Packet-Optical Transport Platform:http://www.cyaninc.com/en/our-solutions/z-series/

[27] Centec Networks V330 OpenFlow Switch:http://www.centecnetworks.com/en/NewsView.asp?ID=232&SortID=48

[28] Transmode TM-Series: http://www.transmode.com/en/products/tm-series

[29] Shimonishi, H. Ochiai, H. Enomoto, N. Iwata, A. 2009. Building Hierarchical Switch Network Using OpenFlow. INCOS'09, 391 – 394.

[30] Jose, L. Yu, M. and Rexford, J. 2011 Online measurement of large traffic aggregates on commodity switches. Hot-ICE'11, USENIX,13-13.

[31] Lu, G. Miao, R. Xiong, Y. Guo, C. 2012. Using CPU as a Traffic Co-processing Unit in Commodity Switches HotSDN'12, ACM.

[32] Sadasivarao, A. Syed, S. Pan, P. Liou, C. Lake A. and Guok, C. 2013 Open Transport Switch - A Software Defined Networking Architecture for Transport Networks. HotSDN'13.

[33] Huang, V K., Yocum, Snoeren, A. C. 2013 High-Fidelity Switch Models for Software-Defined Network Emulation. HotSDN'13, ACM.

[34] Mogul, J. C. and Congdon, P. 2012. Hey you darned counters!: get off my ASIC!. HotSDN '12, ACM. 25-30.

[35] Roy, A. Yocum, K and Snoeren, A. C. 2013. Challenges in the emulation of large scale software defined networks**.** APSys '13, ACM**,** Article No. 10**.**

[36] Vahdat, A. Yocum, K. Walsh, K. Mahadevan, P. Kostic, D. Chase, J and Becker, D. 2002. Scalability and accuracy in a large-scale network emulator. ACM SIGOPS Operating Systems Review - OSDI '02, 271–284.

[37] Lantz, B. Heller, B. and McKeown, N. A 2010. Network in a laptop: rapid prototyping for software-defined network. 9th ACM SIGCOMM Workshop Hotnets-IX, ACM, Article No.19.

[38] Gupta, M. Sommers, J. and Barford, P. 2013. Fast, accurate simulation for SDN prototyping. HotSDN '13, ACM, 31-36.

[39] Z. Chen, Q. Gao, W. Zhang and F. Qin. 2010. FlowChecker: Detecting Bugs in MPI Libraries via Message Flow Checking. IEEE, SC10.

[40] Wundsam, A. Levin, D. Seetharaman, S and Feldmann, A. 2011. OFRewind: enabling record and replay troubleshooting for networks. USENIXATC'11, 29-29.

[41] Handigol, N. Heller, B. Jeyakumar, V. Mazieres, D, McKeown, N. 2013. Where is the Debbuger for my Software Defined Networks. HotSDN'13.

[42] Kazemian, P. Varghese, G. and McKeown, N. 2012. Header Space Analysis: Static Checking for Networks. NSDI'12, USENIX, 9-9.

[43] Canini, M. Venzano, D. Peresini, P. Kostic, D. and Rexford, J. 2012. A NICE way to test OpenFlow applications. NSDI'12, USENIX, 10-10.

[44] Khurshid, W. Zhou, M. Caesar and P. B. Godfrey. 2012. VeriFlow: Verifying Network-Wide Invariants in RealTime. HotSDN'12, ACM, 49-54.

[45] Guha, A. Reitblatt, M. and Foster, N. 2013. Machine-Verified Network Controllers. PLDI'13, ACM, 483-494.

[46] Reitblatt, M, Foster, N., Rexford, J., Schlesinger, C. and Walker, D. 2012. Abstractions for network update. SIGCOMM '12, ACM, 323-334.

[47] Mizrahi T. and Moses, Y. 2013. Time-based Updates in Software Defined Networks. ACM SIGCOMM workshop HotSDN '13, ACM, 163-164.

[48] Vanbever, L., Reich, J. Benson, T. Foster, N. and Rexford, J. 2013. HotSwap: Correct and Efficient Controller Upgrades for Software-Defined Networks. HotSDN'13, Open Networking Summit.

[49] Katta, N. P. Rexford, J. and Walker, D. 2013. Incremental Consistent Updates. SIGCOMM workshop on HotSDN'13, ACM. 49-54.
[50] McGeer, R. A. 2013. Correct, Zero-Overhead Protocol for Network Updates. SIGCOMM workshop on HotSDN'13, ACM, 161-162.
[51] Hinrichs, T.L. Gude, N. S., Casado, M. Mitchell, J. C. and Shenker, S. 2009. Practical declarative network management. WREN'09, 1-10.
[52] Foster, N. Harrison, R. Freedman, M. J. Monsanto, C. Rexford, J. Story A. and D. Walker. 2011. Frenetic: A network programming language. Proceedings of the 16th ACM SIGPLAN international conference ICFP'11, ACM, 279-291.
[53] Voellmy, A. and Hudak, P. 2011. Nettle: Functional reactive programming of OpenFlow networks, PADL.
[54] Monsanto, C. Foster, N. Harrison, R. and Walker, D. 2012. A Compiler and Run-time System for Network Programming Languages. POPL'12, ACM, 217-230.
[55] Voellmy, A., Kim, H. and N. Feamster. 2012. Procera: a language for high-level reactive network control. HotSDN'12, ACM, 43-48.
[56] Katta, N. P. Rexford, J. Walker, D. 2012. Logic Programming for Software-Defined Networks. XLDI.
[57] C. Monsanto, N. Foster, R. Harrison and D. Walker 2012. A Compiler and Run-time System for Network Programming Languages. POPL'12, ACM, 217-230.
[58] Kreutz, D. Ramos, F. M. V. and Verissimo, P. 2013. Towards Secure and Dependable Software-Defined Networks. HotSDN'13, ACM, 55-60.
[59] Reitblatt, M. Canini, M, Guha, A.and Foster, N. 2013. FatTire: declarative fault tolerance for software-defined networks. HotSDN'13, ACM, 109-114.
[60] Kuga, Y. Matsuya, T. Hazeyama, H. Cho, K and Nakamura, O. 2013. EtherPIPE: an ethernet character device for network scripting. ACM SIGCOMM workshop on HotSDN'13, 61-66.
[61] Nelson, T. Guha, A. Dougherty, D. J. Fisler, K. and Krishnamurthi, S. 2013. A balance of power: expressive, analyzable controller programming. HotSDN'13, ACM, 79-84.
[62] OS API: http://www.aristanetworks.com/en/products/eos/software-defined-networking-Arista eAPI Solution Guide
[63] Floodlight Northbound API: http://www.openflowhub.org/display/floodlightcontroller/REST+API
[64] VMware vSphere Management SDK http://www.vmware.com/support/pubs/sdk_pubs.html
[65] VellOS RESTful API: vellosystems.com/wp-content/uploads/pdf/vello-restful-api-guide-1.0.pdf
[66] Quantum API: http://docs.openstack.org/api/openstack-network/1.0/content/
[67] Junos SDK, XML API: http://www.extremenetworks.com/solutions
[68] iRules: http://www.f5.com/products/technologies/irules/
[69] ExtremeXOS InSite SDK API: http://www.extremenetworks.com/solutions/Insite.aspx
[70] Open Automation Framework API: http://www.force10networks.com/CSPortal20/KnowledgeBase/DOCUMENTATION/CLIConfig/FTOS/Automation_2.2.0_4-Mar-2013.pdf
[71] Open Networking Environment Platform Kit (onePK) Nexus 1000V XML API: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9902/white_paper_c11-728045.html
[72] ADX OpenScript API: www.brocade.com/.../B.../ServerIron_12400a_OpenScript_API.pdf
[73] SDN API Management Apigee: http://apigee.com/about/api-best-practices/api-management-software-defined-network-sdn