Journal of Network and Computer Applications 67 (2016) 1–25

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Software defined networks: A survey

Rahim Masoudi, Ali Ghaffari *

Department of Computer Engineering, Tabriz branch, Islamic Azad University, Tabriz, Iran

ARTICLE INFO

Article history:
Received 2 December 2015
Received in revised form
14 February 2016
Accepted 22 March 2016
Available online 26 March 2016

Keywords:
SDN (software-defined network)
OpenFlow
Network virtualization
Network security
QoS (quality of service)
Control plane
Data plane
Programmable networks

ABSTRACT

As a result of the development of internet and ICT (information-centric technology) advances including mobile, cloud, social networking, big data, multimedia and the tendency towards digital society, the management and configuration of them have become highly complex, challenging and time consuming. Also, access to high bandwidth, extendibility and dynamic management are of critical significance, especially when network devices are vertically integrated. Hence, a set of unique predefined line commands and operating systems or firmware should be used. SDN (software-defined networking) is a structure designed for simplifying and improving network management with high flexibility by splitting control plane and data plane. Thus, network programmability is enhanced which in turn leads to more innovation opportunities. Although SDN is regarded as a new research issue, it has attracted numerous researchers' attention from both industrial and academic institutes. In this paper, data plane, control plane and application plane as the three planes of SDN and the interfaces between them such as OpenFlow are investigated and the challenges and the latest technologies in relation to SDN are examined. The investigation and overview of SDN reported in this paper might be used by the interested future researchers to better understand and apply SDN in real-life applications.

© 2016 Elsevier Ltd. All rights reserved.

Contents

* Corresponding author.
E-mail addresses: stu.rmasoudi@iaut.ac.ir (R. Masoudi), A.Ghaffari@iaut.ac.ir (A. Ghaffari).

http://dx.doi.org/10.1016/j.jnca.2016.03.016
1084-8045/© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

As a result of the development of internet and ICT (information-centric technology) advances including mobile, cloud, social networking, big data, multimedia and the tendency towards digital society, the management and configuration of them have become highly complex, challenging and time consuming. Also, access to high bandwidth, extendibility and dynamic management are of critical significance, especially when network devices are vertically integrated. Hence, a set of unique predefined line commands and operating systems or firmware should be used. SDN (software-defined networking) is a structure designed for simplifying and improving network management with high flexibility by splitting control plane and data plane. Thus, network programmability is enhanced which in turn leads to more innovation opportunities. Although SDN is regarded as a new research issue, it has attracted numerous researchers' attention from both industrial and academic institutes. In this paper, data plane, control plane and application plane as the three planes of SDN and the interfaces between them such as OpenFlow are investigated and the challenges and the latest technologies in relation to SDN are examined. The investigation and overview of SDN reported in this paper might be used by the interested future researchers to better understand and apply SDN in real-life applications.

Internet structure and computer networks usually consist of different network devices such as router, switch and different types of middle-boxes which are vertically-integrated and designed by chips and ASIC (application-specific integrated circuits) with high throughput and a specific function. For managing and configuring such network devices, a set of specific and predefined line commands based on embedded operating system is used. Hence, it can be argued that managing a large number of network devices is a big challenge which is prone to many errors. Thus, traditional networks are hardware-centric which suffer from significant shortcomings regarding research and innovations, reliability, extensibility, flexibility and flexibility and manageability. Since internet and mobile networks develop and new technologies such as cloud, social networking and virtualization emerge, the need for networks with higher bandwidth, higher accessibility and dynamic management is becoming a critical issue.

For solving the problems and limitations of traditional networks, a structure, known as SDN, was proposed where network control is split from the forwarding mechanism and it can be programmed and controlled directly. SDN uses a controller which is logically centralized and has a global view towards the network and several simple packet forwarding devices (SDN switches) are controlled and configured through interfaces such as ForCES and Open-Flow. SDN switches are made up of one or more forwarding tables which are controlled by the centralized controller. In other words, they are controlled and programmed in the control plane. Using this mechanism, software developers can easily control network resources. Also, packets are handled by forwarding tables. That is, with respect to the policies accomplished by the centralized controller on forwarding tables, SDN switches can operate in the same way as router, switch, NAT, firewall, etc. Splitting control plane and data plane simplifies the management of modern networks and provides the opportunity for more innovations. As a result, researchers can test and investigate their own ideas and evaluate the results. Since SDN plays significant roles in modern internet structure and ICT technology, it has attracted researchers' attention.

SDN structure consists of three main parts. At the lowest level,

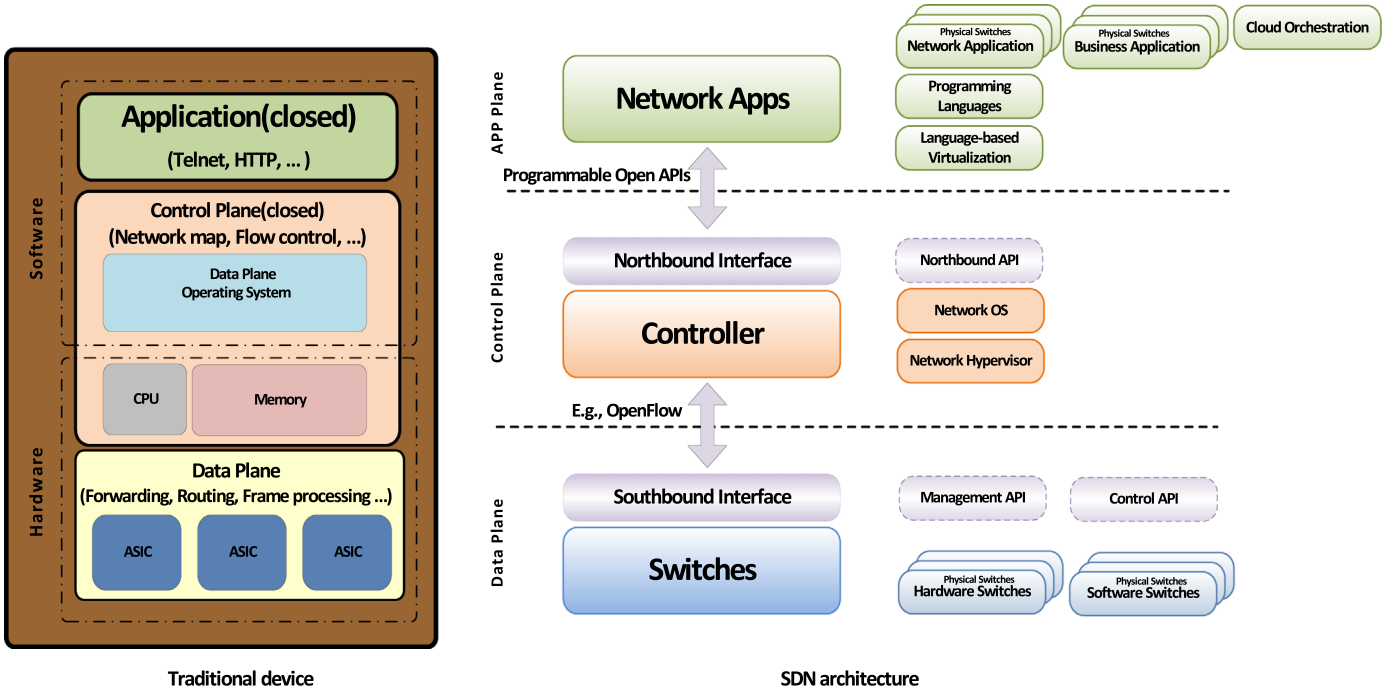

Fig. 1. Component of traditional device and SDN structure.

it includes data plane. At the highest level, it has the application plane and the control plane is between them. The communication between controllers and data plane is maintained via SBI (southbound interface) which is located in SDN switches and the communication between applications and controllers is maintained by NBI (northbound interface) which is located in the control plane (Fig. 1). Using the split between control and data planes, applications follow their own particular purpose such as security method, QoS, traffic engineering and solutions for network measurement and monitoring. Furthermore, controller helps applications to reach their purpose by controlling SDN switches through forwarding tables. In other words, network adjusts itself to users' needs and, using controller and API (application program interface), network managers can easily control the network automatically by adding new features to the control plane without making changes in the data plane.

In this paper, using a comprehensive knowledge and understanding of the key concepts of SDN, the researchers focused on the latest investigations and findings on data, control and application planes and tried to address the hottest and most challenging issues in this domain. Nevertheless, many previous studies focused on only one plane. One of the objectives of this study was to provide an overview on SDN so that research gaps can be highlighted and examined in the future studies. The remaining sections of this paper are organized in the following way. In section two, the related works are briefly reviewed. Then, in sections three, four and five, different classifications of SDN, i.e. application plane, control plane and data plane are discussed, respectively and the related technologies are shortly described. In section six, the challenges and future works on SDN are given and described. Finally, in section seven, the conclusion to the study is given.

2. Related works

In this section, we present an overview of early SDN-related survey efforts. In (Nunes et al., 2014), the authors provided an overview of history of programmable networks from early ideas until recent developments, also they described in detail the SDN paradigm and architecture and the OpenFlow (McKeown et al., 2008) standard that laid the foundation for many of the ideas we are seeing today. And current SDN implementations and testing platforms, applications and examined network services that have been developed based on the SDN paradigm were presented. A comprehensive survey of the important topics in SDN/OpenFlow implementation, including the basic concept, applications, language abstraction, controller, virtualization, quality of service, security issues, and its integration with wireless and optical networks in Hu et al. (2014a) were surveyed. In Xia et al. (2014), the definition of SDN and highlighted benefits of SDN in offering enhanced configuration, improved performance, and encouraged innovation were presented. Moreover, a literature survey of recent SDN researches in the infrastructure layer, the control layer, the application layer and OpenFlow were provided. Farhady et al. (2015) tried to review SDN-related technologies and cover three main parts of SDN: applications, the control plane, and the data plane. The authors argue that data plane programmability needs to be considered in the SDN definition. They provide data plane technologies and discuss several future directions to realize data plane programmability. And they believe that current southbound APIs are not flexible and are mostly translated as OpenFlow. If we can define a better southbound API, then we can add interesting operation and management features and facilities to the data plane. The authors in Macedo et al. (2015) presented the state of-the-art in network programmability, highlighting the key technologies on both data plane and control plane programmability. They categorize a broad set of technologies, ranging from software-defined networks, virtualization, software-defined radios as well as full software data plane implementations, showing that they are complementary tools that can be used to build programmable networks. Traffic engineering (TE) is an important mechanism to optimize the performance of a data network by dynamically analyzing, predicting, and regulating the behavior of the transmitted data. The authors in Akyildiz et al. (2014) provide an overview of traffic engineering mechanisms in SDN architectures. They study the traditional traffic engineering

technologies from early ideas in ATM networking through current developments in IP and MPLS networking. Also, availability, scalability, reliability, and consistency in data networking with SDN, the traffic management with regard to load balancing, fault tolerance, consistent network update methods, as well as traffic analysis for testing and debugging network systems and network monitoring tools were investigated.

In Kobayashi et al. (2014), the authors report on early deployments of SDN on university campuses. The deployments, combined with the applications and experimentation, played an important role in demonstrating the SDN potential, and gain the community's mindshare. They also demonstrated the key value proposition of SDN; proved that the physical infrastructure can support multiple concurrent virtual networks for research and production use; revealed a number of performance tradeoffs; provided valuable input to the OpenFlow specification process; and helped vendors and help grow the larger ecosystem.

Recent and state-of-the-art projects in SDN were surveyed in Hakiri et al. (2014) and have been classified key challenges and opportunities along a number of different areas including architectural models, programmability, convergence, wireless and mobility, cloud platforms, and security. It shows that the networking community is heavily involved in SDN research, however, most of the investigations continue to focus on topics such as control plane/data plane, distributed vs. centralized control plane, scalability of solutions, Hybrid solutions, call graph, networking models etc. So, while these research efforts are important, they need to occur in the context of overall network programmability and scalability goals.

Wireless networking has been one of the most important and rapidly growing revolutions in recent years, changing everyone's lives and giving people access to the Internet anywhere and anytime. The state of the art of SDN in the context of wireless networks and published research literature on the application of SDN ideas in wireless networks in Jagadeesan and Krishnamachari (2014) were surveyed. It begins with a brief overview of the history of SDN and what the term entails. It also presents an overview of the major design trends and highlight key differences between them. The SDN concept of centralized control was extended to wireless distributed networks (WDNs). In Abolhasan et al. (2015) a new SDN architecture for WDNs was presented, which eliminates the need for multi-hop flooding of route information and therefore enables WDNs to easily expand. The key idea is to split network control and data forwarding by using two separate frequency bands. The forwarding nodes and the SDN controller exchange link-state information and other network control signaling in one of the bands, while actual data forwarding takes place in the other band.

Existing commercial wireless networks are inherently hardware-based and rely on closed and inflexible architectural designs. SoftAir (Akyildiz et al., 2015) as a new paradigm towards next-generation (5G) wireless networks was proposed. SoftAir is high flexible architecture, which can accelerate the innovations for both hardware forwarding infrastructure and software networking algorithms through control and data plane separation, enable the efficient and adaptive sharing of network resources through network virtualization, achieve maximum spectrum efficiency through cloud based collaborative baseband processing, encourage the convergence of heterogeneous networks through open and technology independent interfaces, and enhance energy efficiency through the dynamic scaling of computing capacity of the software-defined base stations (SD-BSs). Moreover, SDN concept can be applied to wireless mesh network that has been widely adopted by various applications. A novel architecture of software-defined wireless mesh networks (SD-WMNs) (Huang et al., 2015) providing Internet services was proposed. Since wireless spectrum is a scarce resource that is shared by both data and control traffic in SD-WMNs, three novel spectrum allocation and scheduling algorithms, namely FB-NS(fixed-bands non-sharing algorithm), NFB-NS(non-fixed-bands non-sharing algorithm), and NFB-S (non-fixed-bands sharing algorithm) that orchestrate both control and data traffic was proposed.

Different from the concept of decoupling the control and data planes in SDN, network functions virtualization (NFV) aims to introduce and deploy new network functions to provide flexible management using server virtualization techniques without specialized hardware in an open and standardized information technology (IT) environment. So, both SDN and NFV are effective approaches to mitigate the challenges of legacy networks. Cao et al. (2015) reviews the concepts of SDN and WNV, and then design a SDN based approach to realize WNV, called software-defined virtual wireless network (SDVWN).

The SDN architecture can be exploited to enhance network security with the provision of a highly reactive security monitoring, analysis and response system. The logical centralization of network intelligence presents exciting challenges and opportunities to enhance security in such networks, including new ways to prevent, detect and react to threats, as well as innovative security services and applications that are built upon SDN capabilities. Scott-Hayward et al. (2013) and Ali et al. (2015) present a comprehensive survey of recent works that apply SDN to security, and identify promising future directions that can be addressed by such research. The authors in Ali et al. (2015) have classified their work in two main streams threat detection, remediation and network correctness (which simplify and enhance security of programmable networks), and security as a service (which offers new innovative security functionality to users, such as anonymity and specialized network management).

3. Data plane

Review related-SDN works from the beginning of SDN research, focuses are more on the development and programmability of the control plane. So we need data plane focused research in addition to control plane for SDN. In this section, we review data plane related contributions in SDN to indicate there is a gap that need to be considered from the community. Next, were presented some existing technologies that can be used to realize a software-centric SDN data plane compared with the current hardware-centric proposals; and finally interfaces between data plane and control plane.

Packet forwarding is one of the basic and primary functions of the data plane. In addition, data plane programmability enables various functions such as network appliances (e.g., for deep packet inspection), in-network processing (e.g., cache and transcoding). Furthermore, networking tasks such as anomaly detection and traffic engineering depend on either customized hardware or special protocols like IP. So, data plane programmability may be able to satisfy those requirements: First, supporting a new protocol needs a change in the data plane and second, flexibility in adapting to new protocols and architectures rely on an open infrastructure.

3.1. Packet forwarding infrastructure

Similarly to a traditional network, a SDN infrastructure is composed of a set of networking equipment (switches, routers and middle-box appliances). In SDN, forwarding elements are simple without embedded control or software to take autonomous decisions. Network intelligence and state are logically centralized, and the underlying network infrastructure (switches, vSwitches,

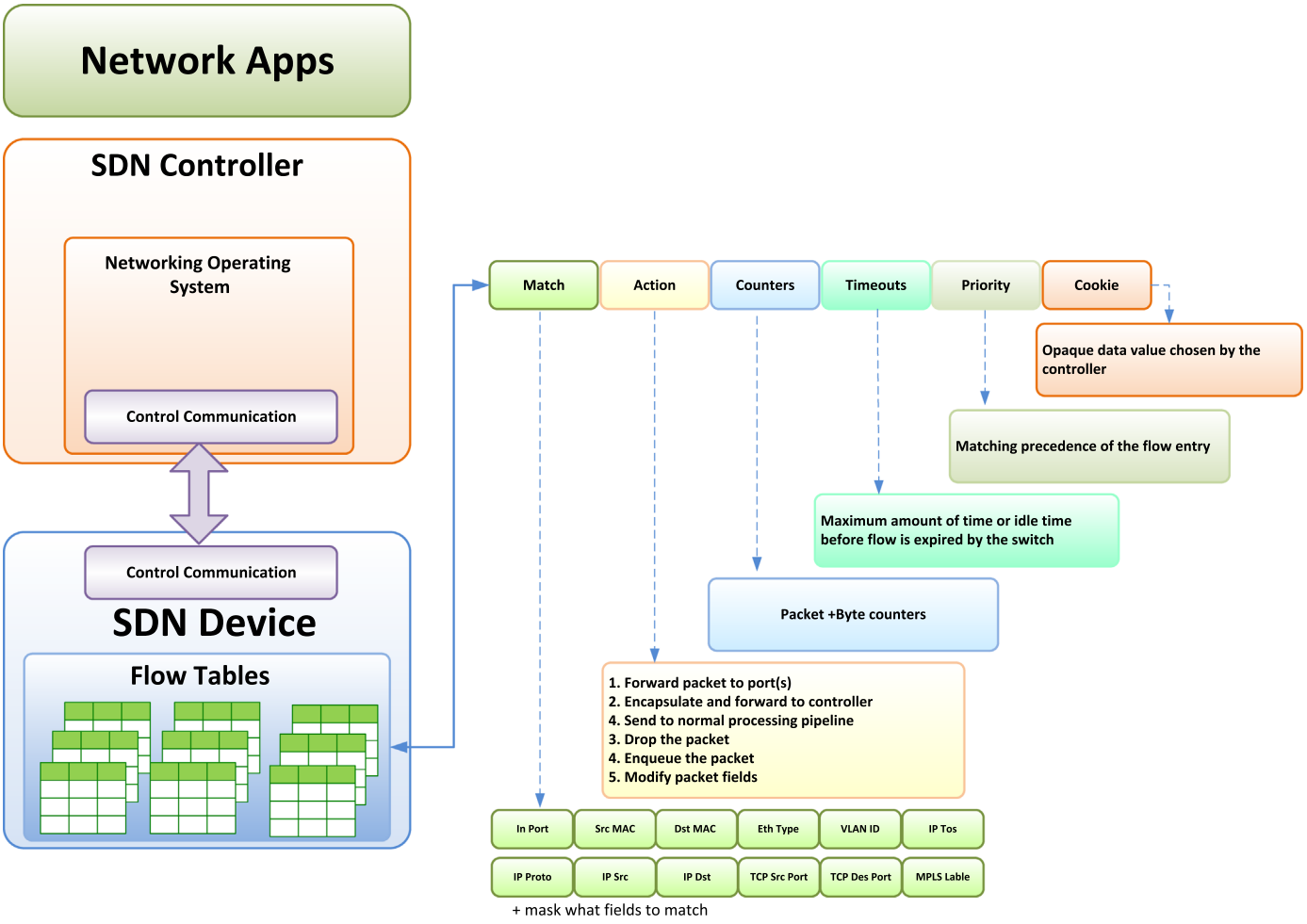

Fig. 2. OpenFlow-enabled SDN devices.

routers, etc.) is abstracted from the applications. OpenFlow provides an open standards interface to vendor network devices. The network devices can be programmed by the SDN controller using the OpenFlow protocol. There are two main elements in SDN/OpenFlow architecture, the controllers and the forwarding devices, as shown in Fig. 2. A packet is forwarded by the switches based on the entries in the flow table. The flow entries of flow tables can be updated, deleted, and added by Controller with using OpenFlow messages. OpenFlow switches possess a much simpler flow table than ordinary switches. The flow table in the OpenFlow switch consists of many flow entries, each of which includes six parts (Fig. 2). The Match fields is used to match against packets, which consists of ingress port and packet headers; the Priority is adopted for matching the precedence of the flow entry; the Counters is used to update for matching packets; the Instructions is used to modify the action set or pipeline processing; the Timeouts sets the maximum amount of time or idle time before flow is expired by the switch; and the Cookie is the opaque data value chosen by the controller, which can be used by the controller to filter flow statistics, flow modification, and flow deletion. The Match field supports 12 header fields.

3.2. *OpenFlow switch*

An OpenFlow switch is a software program or hardware device that forwards packets in a software-defined networking (SDN) environment. An OpenFlow switch consists of at least three main parts: (i) *flow table*: is used to lookup packet and also do forwarding, (ii) *secure channel*: is usually a TLS or SSL channel between switch and controller, (iii) *OpenFlow protocol*: is for communicating with the switches and managing them. OpenFlow switches are either based on the OpenFlow protocol or compatible with it (Fig. 3).

3.2.1. *Software switches*

Open source software switch aims to implement a switch platform in virtualized server environments. It supports standard management interfaces and enables programmatic extension and control of the forwarding functions. There are currently several SDN software switches available that can be used, for example, to run a SDN testbed or when developing services over SDN. Table 1 summarizes a list of current software switch implementations with a brief description including implementation language and the OpenFlow standard version that the current implementation supports.

3.2.2. *Hardware switches*

The OpenFlow standard is one of the main SDN enabling technologies currently being implemented in commodity hardware networking. Table 2 gives a brief list of commercial switches that are currently available in the market, status about OpenFlow version and their Maker.

3.3. *Data plane technologies*

In this section we review current efforts towards developing

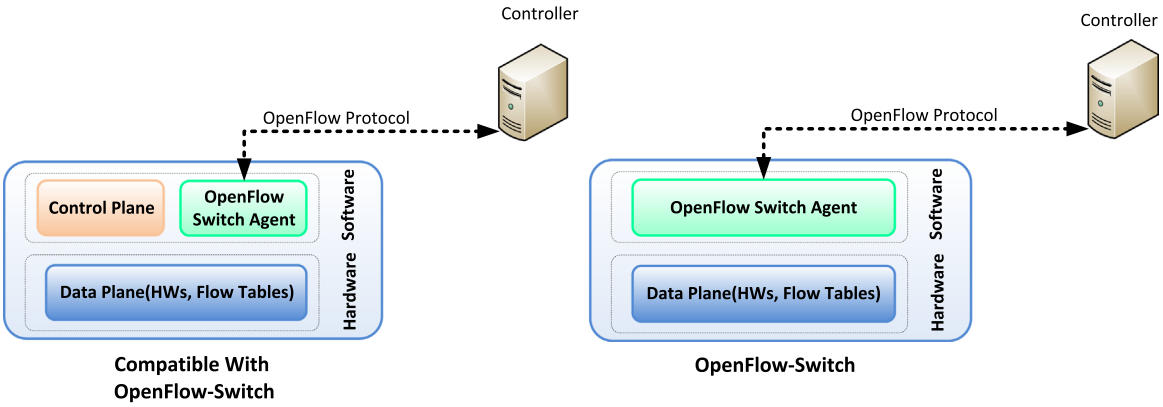

Fig. 3. OpenFlow switch.

the SDN data plane which are mostly OpenFlow-centric and software-centric SDN data plane

3.3.1. *OpenFlow-centric SDN data plane*

There are a few researches that show two limitations in current switching chips and the OpenFlow protocol: (i) current hardware switches are quite rigid, allowing "Match-Action" processing on only a fixed set of fields (ii) The OpenFlow specification only defines a limited repertoire of packet processing actions. In Bosshart et al. (2013) the authors propose the RMT (reconfigurable match tables) model, a new RISC-inspired pipelined architecture for switching chips, which allows the forwarding plane to be changed in the field without modifying hardware. Also, RMT allows the programmer to modify all header fields much more comprehensively than in OpenFlow. The authors in Sivaraman et al. (2013) argue the way forward requires carefully extending SDN to control the fast-path scheduling and queueing behavior of a switch. They propose adding a commodity/programmable hardware such as FPGA (field programmable gate arrays) to the data plane to enable SDN programmability extended. In Bianco et al. (2010), the authors focus on the data path and analyze the OpenFlow implementation in Linux based PCs. They compare OpenFlow switching, layer-2 Ethernet switching and layer-3 IP routing performance. Forwarding throughput and packet latency in under-loaded and over-loaded conditions are analyzed, with different traffic patterns. System scalability is analyzed using different forwarding table size, and fairness in resource distribution is measured.

A complementary design approach to OpenFlow's conventional designs was proposed in Luo et al. (2009), which used network processor based acceleration cards to perform OpenFlow switching. It shows a 20% reduction on packet delay and the comparable packet forwarding throughput compared to conventional designs. The authors in Jeyakumar et al. (2013), show how end-hosts can coordinate with the network to implement a wide-range of network tasks, by embedding tiny programs into packets that execute directly in the data plane. The key contribution is a programmatic interface between end-hosts and the switch ASICs that does not sacrifice raw performance. This interface allows network tasks to be refactored into two components: (i) a simple program that executes on the ASIC; (ii) an expressive task distributed across end-hosts. The authors in Tanyingyong et al. (2010) propose an architectural design to improve lookup performance of OpenFlow switching in Linux using a standard commodity network interface card based on the Intel 82599 Gigabit Ethernet controller. It shows packet switching throughput increasing up to 25 percent compared to the throughput of regular software-based OpenFlow switching. In Kogan et al. (2013), the authors presented research directions that can significantly reduce TCAM (ternary-content-addressable memory) and control plane requirements via classifier sharing and reuse of existing infrastructure elements. They show how to generalize virtual pipeline architecture to distribute workload which, OpenFlow can be extended to distribute processing.

3.3.2. *Software-centric SDN data plane*

The most basic and fundamental requirement for a software-centric data plane, is Software based packet switching and forwarding. In this section, we review many proposals for software forwarding plane that focus on the performance aspects of the research using different underlying commodity hardware such as CPU, GPU, NPU (network processing unit), and FPGA. In Dobrescu

Table 1
OpenFlow-related software projects switches.

Product	Maker/Developer	Type	Description	Version
Contrail-router (Juniper/contrail-vrouter, 2016)	Juniper Networks	vrouter	The Contrail Virtual Router implements the data-plane functionality that allows a virtual interface to be associated with a VRF.	1.0
LINC (FlowForwarding/LINC-Switch, 2016)	FlowForwarding	switch	LINC is a pure OpenFlow software switch which is implemented in operating system's user space as an Erlang node.	1.4
ofsoftswitch13 (CPqD/ofsoftswitch13, 2016)	Ericsson, CPqD	switch	Ofsoftswitch13 is an OpenFlow 1.3 compatible user-space software switch implementation.	1.3
Open vSwitch (Open vSwitch, 2016)	Open Community	switch	Open vSwitch is a network switch specifically built for virtual environments.	1.0–1.3
OpenFlow Reference (OpenFlow, 2016)	Stanford	switch	OpenFlow Switching capability to a Linux PC with multiple NICs.	1.0
OpenFlowClick (OpenFlowClick, 2016)	Yogesh Mundada	vrouter	OpenFlow switching element for Click software routers.	1.0
OpenFlowJ (floodlight/loxigen, 2016)	Stanford		Source code implementation of OpenFlow protocol. Both Beacon and FlowVisor incorporate this code.	1.0–1.3
OpenFaucet (rlenglet/openfaucet, 2016)	Midokura		Source code implementation of OpenFlow protocol 1.0, used in both controllers and switches.	1.0
Pantou/OpenWRT (Pantou, 2016)	Stanford	switch	Turns a wireless router into an OF-enabled switch.	1.0
Switch Light (Switch Light, 2016)	Big Switch	switch	Thin switching software platform for physical/virtual switches.	1.0
XorPlus (Pica8 Xorplus, 2016)	Pica8	switch	Switching software for high performance ASICs.	1.0

Table 2
OpenFlow-compatible commercial switches.

Product	Maker/Developer	Type	Description	Version
8200zl and 5400zl (HP, 2016)	Hewlett-Packard	Chassis	Data center class chassis (switch modules).	1.0
Arista 7150 Series (Networks A. Arista Networks, 2016)	Arista Networks	Switch	Data centers hybrid Ethernet/OpenFlow switches.	1.0
BlackDiamond X8 (Extreme Networks, 2016)	Extreme Networks	Switch	Cloud-scale hybrid Ethernet/OpenFlow switches.	1.0
CX600 Series (Huawei Technologies Co., Ltd., 2016)	Huawei	Router	Carrier class MAN routers.	1.0
EX9200 Ethernet (Juniper Networks, 2016)	Juniper	Chassis	Chassis based switches for cloud data centers.	1.0
EZchip NP-4 (EZchip, 2016)	EZchip Technologies	Chip	High performance 100-Gigabit network processors.	v1.1
MLX Series (Brocade MLX Series, 2016)	Brocade	Router	Service providers and enterprise class routers.	1.0
NoviSwitch1248 (Networking and Systems, 2016)	NoviFlow	Switch	High performance OpenFlow switch.	1.3
NetFPGA (Nec, 2016)	NetFPGA	Card	1 G and 10 G OpenFlow implementations.	1.0
RackSwitch G8264	IBM	Switch	Data center switches supporting Virtual Fabric and OpenFlow.	1.0
PF5240 and PF5820 (Nec, 2016)	NEC	Switch	Enterprise class hybrid Ethernet/OpenFlow switches.	1.0
Pica83920 (Pica8, 2016)	Pica8	Switch	Hybrid Ethernet/OpenFlow switches.	1.0
Plexxi Switch 1 (PlexxiInc. Plexxi, 2016)	Plexxi	Switch	Optical multiplexing inter connect for data centers.	1.0
V330 Series (Centec Networks, 2016)	Centec Networks	Switch	Hybrid Ethernet/OpenFlow switches.	1.0
Z-Series (Z-Series Packet-Optical Transport Platforms, 2015)	Cyan	Switch	Family of packet-optical transport platforms	1.0

et al. (2009), the authors propose a software router architecture that parallelizes router functionality both across multiple servers and across multiple cores within a single server. By exploiting parallelism, a 35 Gbps parallel router prototype was demonstrated; its capacity can be linearly scaled through the use of additional servers. PacketShader (Han et al., 2011) is a high-performance software router framework for high performance network packet processing with graphics processing unit (GPU) acceleration. It minimizes per-packet processing overhead in network stack and performs packet processing in the user space without serious performance penalty. So, a well-designed PC-based router can achieve 40 Gbps forwarding performance with full programmability even on today's commodity hardware. FLARE switch (Nakao, 2012) is a programmable switch using Click environment and multicore CPUs. It has a couple of SFP+ ports and provides a Linux and Click environment for network research. It uses many core NPUs to run the packet forwarding and processing routines in concurrent manner. In Qi et al. (2010), the authors present an FPGA-based architecture to support 100 Gbps packet classification, which based on HyperSplit, a memory-efficient tree search algorithm. This approach uses significantly less FPGA resources; Compared to state-of-the-art FPGA based solutions and can support over 50 K rules with a single FPGA chip. Also, Compared to multi-core based solutions, this approach has at least a 10 × speedup. L7Classifier (ANDO and NAKAO, 2014) is a packet classification method based on the packet payload (i.e., application layer) compared with traditional methods that use L2–L4 header information. It stores TCP flow information and performs regular expression matching to packet payload.

Most routers or switches have closed, static, and inflexible designs. Network administrators cannot easily implement new functions, specify or even identify the interactions of different functions, but they may be able to turn router functions on or off. Extensions require access to software interfaces in the router's forwarding path, but these often do not exist, do not exist at the right point, or are not published. Click (Kohler et al., 2000) is a new software architecture for building flexible and configurable routers. It is assembled from packet processing modules called elements. Individual elements implement simple router functions like packet classification, queueing, scheduling, and interfacing with network devices. The authors in Ahmed et al. (2012) presented ClickOS to making the data plane more programmable. ClickOS is a tiny network operating system based on Xen and the Click modular router system which can run a wide range of middle-boxes. So, it can be used add and remove features very fast and dynamic. vNode (Nakao, 2012) is a Click-like programming environment to the user in order to enable implementing new features and requirements easier and faster with network virtualization technology.

Current SDN/OpenFlow data plane does not allow statefull processing of packets within the switch box. OpenState (Bianchi et al., 2014) proposes a first step in the direction of supporting statefull per flow processing over closed platforms. It offers a viable abstraction based on extended finite state machines as an extension (super-set) of the OpenFlow match/action abstraction. SP4 (Gill et al., 2012) is a software-based programmable packet processing platform that supports statefull packet processing useful for analyzing traffic flows with session semantics. It are used for performing high-throughput analysis of traffic traces for a variety of applications, such as filtering out unwanted traffic and detection of DDoS attacks using machine learning based analysis. In the similar way, the authors in (Farhadi et al., 2014), extend the programmability and flexibility of SDN to the data plane to allow network owners to add their custom network functions while keeping the programmability of existing SDN. As an example, NetOpen switch node (Kim et al., 2012) supports customized in-network processing and verify its programmability and in-network processing performance for multiple flows (also (Nishida and Nakao, 2012), (Lee et al., 2011) and (Shimamura et al., 2013)).

There are several security threats to OpenFlow, which introduced in (Benton et al., 2013); and some data plane and control plane mechanisms can provide security for end-points and users. FRESCO (Shin et al., 2013), is an OpenFlow security application development framework designed to facilitate the rapid design, and modular composition of OpenFlow-enabled detection and mitigation modules. It introduces minimal overhead and enables rapid creation of popular security functions with significantly (over 90%) fewer lines of code. FRESCO offers a powerful new structure for prototyping and delivering innovative security applications into the rapidly evolving world of SDNs. In Jang et al. (2011) the authors offer high-performance SSL (secure sockets layer) acceleration using commodity processors. They show that modern GPUs can be easily converted to general-purpose SSL accelerators. By exploiting the massive computing parallelism of GPUs, SSL cryptographic operations beyond what state-of-the-art CPUs provide was accelerated.

3.4. Southbound interface

Southbound interfaces are the APIs that enables communications between the control plane and the data plane. OpenFlow protocol is the most well-known interface between forwarders and controllers in SDN. OpenFlow is an SDN technology proposed to standardize the way and defines an API for the communication

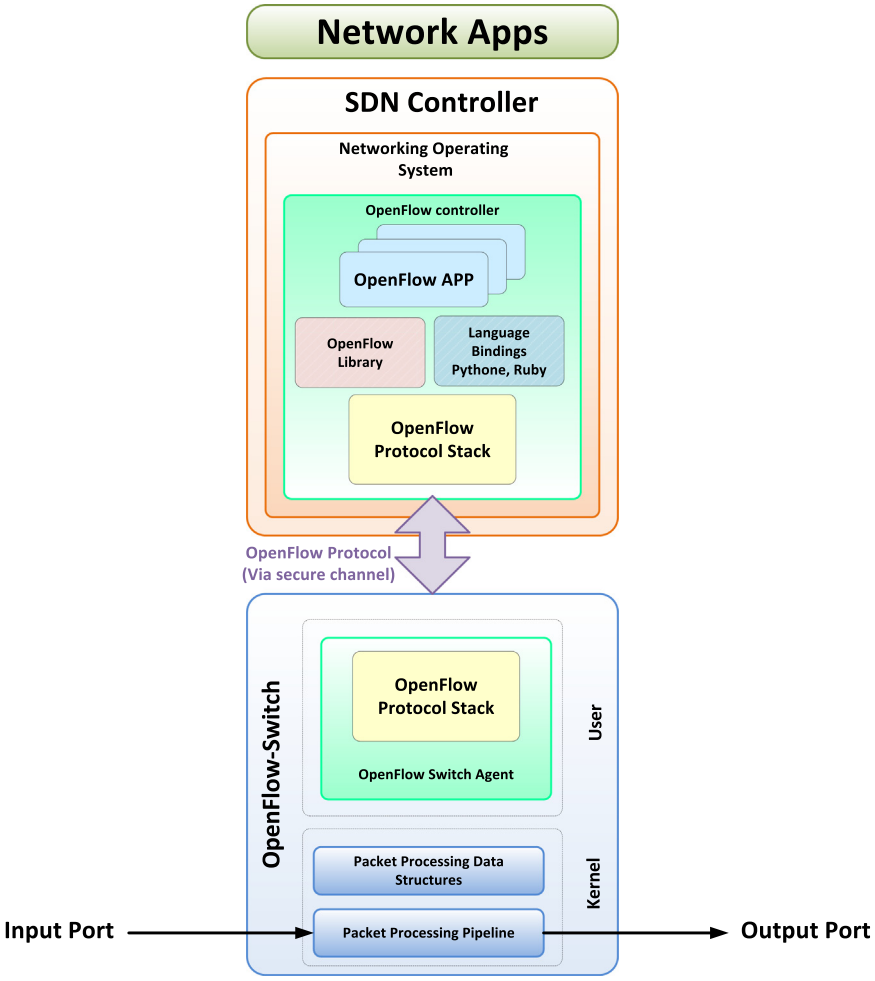

Fig. 4. OpenFlow architectural components.

between the controller and the switches (Fig. 4). It provides a specification to migrate the control logic from a switch into the controller. Therefore, both the controller and the switches should understand the OpenFlow protocol. OpenFlow-based architectures have specific capabilities that can be exploited by researchers to experiment with new ideas and test novel applications. These capabilities include software-based traffic analysis, centralized control, dynamic updating of forwarding rules and flow abstraction. OpenFlow-based applications have been proposed to ease the configuration of a network, to simplify network management and to add security features, to virtualize networks and data centers and to deploy mobile systems. The authors in Suzuki et al. (2014) presented a survey on OpenFlow related technologies that have been offered as a means for researchers, network service creators, and others to easily design, test, and deploy their innovative ideas in experimental or production networks to accelerate research activities on network technologies.

OpenFlow protocol has different versions and features of each version are as follows.

- *Version 0.2.0:* released in March 2008 as a draft.
- *Version 1.0:* specification is the first version which has official vendor support (Consortium, 2016). It supports a single flow table with flow entries consist of three components: Header Fields, Counters and Actions.
- *OpenFlow 1.1:* was released on February 28, 2011, which introduced multiple tables pipeline processing (Agarwal et al., 2013).
- *Version 1.2:* In December 2011, the ONF board approved OpenFlow version 1.2 and published it in February 2012, which added support for IPv6 (OpenFlow Switch Consortium, 2016). With OpenFlow 1.2 the switches can connect to multiple controllers concurrently. This allows for a better failure recovery and also loads balancing between controllers.
- *Version 1.3:* It provides more support for protocols such as MPLS BoS bit and IPv6 extension headers. It also includes a better expression of switch capabilities (i.e., refactor capabilities negotiation) and improves metering facilities (e.g., per flow metering) (OpenFlow Switch Consortium, 2016).
- *Version 1.4:* It improves the OpenFlow extensible match introduced in v1.2 that gives more flexible classification capabilities to the user to match packet header fields and classify flows (OpenFlow Switch Consortium, 2016).
- *Version 1.5:* The next version of the main branch, version 1.5, is being worked on and planned for late 2014 (OpenFlow Switch Consortium, 2016).The set of features in 1.5 is not finalized; some of the features may not survive the specification process. Some example features planned for 1.5 are delegation of learning, egress table, flexible statistics, matching on TCP flags, a copy-field action and support for optical switches.

OpenFlow is not the only available southbound interface for SDN; there are also other southbound interfaces such as Open vSwitch database management protocol (OVSDB) (Pfaff and Davie, 2013), forwarding and control element separation (ForCES) (Wang et al., 2010), protocol oblivious forwarding (POF) (Song, 2013),

OpFlex control protocol (Smith et al., 2014), OpenFlow Config (OF-Config) (ONF, 2014), OpenState (Bianchi et al., 2014), revised OpenFlow library (ROFL) (Sune et al., 2014), hardware abstraction layer (HAL) (Parniewicz et al., 2014) and Programmable Abstraction of Data path (PAD) (Belter et al., 2014a).

4. Control plane

The control plane in SDN is called a controller; it acts as an intermediary layer between applications by northbound Interface (NBI) and the data plane by southbound interface (SBI). A controller plays a main role like a brain and provides an abstract and centralized view of the overall network. Each control plane is built from two components; applications and network operating system (NOS). The application part is in many of software programs from metering to monitoring which network virtualization is one of them and NOS acts as a SDN controller. So, in the control plane, it is significant to design interfaces and the controller itself in an effective way. Network could have more than one controller, so each controller is responsible to control a group of network switches, which may interfere with each other, thus one Controller is chosen to be the main controller and the others would be backups. In this section, we review existing efforts, particularly OpenFlow-related research, to realize and improve various aspects of the control plane; because OpenFlow is prominently successful, whereas other approaches (e.g. ForCES) are not as successful in practice.

4.1. Northbound interfaces

Interfaces between SDN Applications and Controllers are SDN northbound interfaces and typically provide abstract network views and enable direct expression of network behavior and requirements. NBI presents a programmable application programming interface (API) to network control and management applications. The NBIs and SBIs are two key abstractions layer of the SDN system. All controller solutions today have proprietary APIs for application interfaces. That is, no standard northbound interface exists in reality, although some are attempting to work on this problem such as the Open Daylight Project. But, the southbound interface has already a widely accepted proposal (OpenFlow and ForCES). One value of SDN lies in the expectation that these interfaces are implemented in in an open, vendor-neutral and interoperable way. In architectural overview, the northbound interface is normally drawn at the top of the component it is defined in, hence the name northbound interface. JSON (Crockford, 2006) or Thrift (Apache Thrift - Home, 2016) is examples of a northbound interface.

4.2. Controller designs

In OpenFlow, several switches were managed by a physically or logically centralized controller. So, the controller design is the most important component in the SDN architecture and importantly affects the overall performance of the network. Due to this, there are many researches on enhancing the controller design to improve the performance of various aspects such as State consistency, scalability, flexibility, security, availability, Latency and placement.

4.2.1. State consistency

OpenFlow controllers and SDN-switches need to hold the same forwarding policy for stable forwarding. FlowAdapter (Pan et al., 2013) is an innovative middle layer which converts flow entry polices from the controller flow table pipeline to switch hardware flow table pipeline, so that the same polices can be fitted into different types of hardware. The authors in Kang et al. (2012), proposed a similar technique which offers a set of principles for policy transformation to enable rewriting of polices between multiple switches while preserving the forwarding policy. Palette distribution framework (Kanizo et al., 2013) decomposes large SDN tables into small ones and then distributes them across the network while preserving the overall SDN policy semantics. It is especially important as switch table sizes can become a bottleneck in scaling SDNs. Moreover, it facilitates handling the heterogeneity of switches in the network and the changes of equipment. HOTSWAP (Vanbever et al., 2013) is a system for upgrading SDN controllers in a disruption-free and correct manner. It is a hypervisor (sitting between the switches and the controller) that maintains a history of network events. In Kang et al. (2013) the authors proposed efficient rule-placement algorithms that distribute forwarding policies across general SDN networks while managing rule-space constraints, and show how to support dynamic, incremental update of policies. ONOS (Open Network Operating System, 2016) maintains consistency across the distributed state by providing a network topology database to the controller. The idea of a network information base is also proposed in Koponen et al. (2010) so as to satisfy the state consistency and durability requirements. HyperFlow (Tootoonchian and Ganjali, 2010) is a distributed event-based control plane for OpenFlow and it is logically centralized but physically distributed. HyperFlow provides scalability while keeping the benefits of network control centralization. By passively synchronizing network-wide views of OpenFlow controllers, HyperFlow localizes decision making to individual controllers, thus minimizing the control plane response time to data plane requests. To realize consistent packet processing (Perešíni et al., 2013), the controller uses transactional semantics. The trade-off between update time and rule-space overhead (Katta et al., 2013) and the trade-off between maintaining consistency during configuration updates and the update performance (Mizrahi and Moses, 2013) are also studied.

4.2.2. Scalability

With the increasing popularity of SDN, designing a scalable SDN control plane becomes a critical problem. An effective technique to enhance the scalability is to design distributed architecture of SDN control plane. The scalability issue in specific areas is studied which try to improve the scalability, particularly by reducing the overhead of the centralized controller in several aspects. DevoFlow (Curtis et al., 2011) is a modification of the OpenFlow model which can simplify the design of high-performance OpenFlow switches and enable scalable management architectures. It reduces the number of interactions between the controllers and switches by handling mice flows (i.e. short flows) at the OpenFlow switch and elephant flows (i.e. larger flows) at the controller. Similarly, DIFANE (Yu et al., 2011) is a scalable and efficient solution which attempts to reduce the number of requests to the controller by proactively pushing the flow entries to switches. It partitions rules over the switches, and keeps all traffic in the data plane by selectively directing packets through intermediate switches. Kandoo (Hassas Yeganeh and Ganjali, 2012) is a highly configurable and scalable control plane, which uses two layers of controllers to limit the overhead of frequent events on the control plane for creating a distributed control plane. Most of the frequent events using the state of a single switch are handled by the controllers at the bottom layer and the events that cannot be handled by the controllers at the bottom layer, are handles by a logically centralized controller of the top layer. The authors in Othman and Okamura (2013) proposed a hybrid control model for flow-based SDN that combines both central control and distributed control, in order to provide a more relaxed and scalable

control. They try to reduce the overhead of the central controller by defining new features such as the proactive flows to be activated under certain conditions. AutoSlice (Bozakov and Papadimitriou, 2012) is a distributed hypervisor architecture that can handle a large number of flow table control messages from multiple tenants to support scalable SDN slicing. It enables the maker with the ability to redesign the SDN for different applications while minimizing operator intervention. SDN is a natural platform for network virtualization, but supporting numerous tenants with different topologies and controller applications raises scalability challenges. To handle this problem, the authors in Drutskoy et al. (2013) proposed the FlowN architecture which aims to offer a scalable solution for network virtualization by providing an efficient mapping between virtual and physical networks and by leveraging scalable database systems. To fully utilize the replicated servers in online services, depend on load balancing. The authors in Wang et al. (2011) argue that the controller should exploit switch support for wildcard rules for a more scalable solution that directs large aggregates of client traffic to server replicas. They present several methods that compute wildcard rules that achieve a target distribution of the traffic, and automatically adapt to changes in load-balancing policies without disrupting existing connections. The authors in Yeganeh et al. (2013) deconstructed scalability concerns and discussed several scalability trade-offs in SDN design space. They argue that they are not unique to SDN.

4.2.3. Flexibility and modularity

The mapping between a switch and a controller (statically configured) is a key limitation of the distributed controllers, which may outcome in uneven load distribution among the controllers. There have been few recent attempts to address this issue. ElastiCon (Wang et al., 2011), is an elastic distributed controller architecture, which manages the controller pool that is dynamically expanded or reduced according to traffic conditions. The mechanism dynamically shifts the load traffic through controllers to gracefully handle it. SDN protocols, to support a global view, expose several counters for each flow-table rule. These counters must be maintained by an ASIC in data plane, but ASIC-based counters are inflexible. To realize far more flexible processing of counter-related information, the authors in Mogul and Congdon (2012) proposed software-defined counters that utilize general-purpose CPUs rather than ASIC-based counters. In a similar spirit, to overcome the limitations of the ASIC-based approach, the authors in Lu et al. (2012) used a CPU as a traffic co-processor in the switches to handle not only the control plane but also data plane traffic. XSP (extensible session protocol) (Kissel et al., 2012) is a framework for applications to interface with SDNs in a secure and dynamic manner, within a general and extensible protocol for managing the interaction between applications and network-based services, also between the devices. One of the major factors in managing complexity of any software system such as SDN is modularity. The work in Monsanto et al. (2013) proposed new abstractions for building applications from multiple, independent modules that jointly manage network traffic to ensure that rules installed to perform one task do not override other rules.

4.2.4. Availability

Tolerating and recovering from link and switch failures are basic requirements of most networks, such as SDN. So, the OpenFlow controller and switches should be powerful in various situations. The authors in Kuźniar et al. (2013), propose a runtime system that automates failure recovery and enables network developers to write simpler, failure-agnostic code by spawning a new controller instance that runs in an emulated environment. Then, it quickly replays inputs observed by the controller before the failure occurred. Finally, it recovers the network by installing the difference rule set between emulated and current forwarding states. The efficient solutions was presented in Kozat et al. (2013) which controller can install static rules on the switches to verify topology connectivity and locate link failures based on these rules. To discover link and node failures and trigger restoration actions, the centralized controller can use link-layer discovery protocol (LLDP) messages. This monitoring and recovery mechanism has serious scalability limitations because the controller has to be involved in the processing of all of the LLDP monitoring messages. To overcome this issue and fast recovery is needed to frequent issuing of monitoring messages, but this may place a significant load on the controller. In Kempf et al. (2012), the authors propose to implement a monitoring function on OpenFlow switches, which can emit monitoring messages without posing a processing load on the controller. RuleBricks (Williams and Jamjoom, 2013) is a system for flexibly embedding high availability support in existing OpenFlow policies by presenting three key primitives: drop, insert, and reduce. In this work, the authors discuss how these primitives can express various flow assignments and backup policies demonstrating the one offered by the Chord protocol.

4.2.5. Security and dependability

The OpenFlow architecture involves third-party development efforts, and therefore suffers from potential trust issue on OpenFlow applications. The misuse of such trust could lead to various types of attacks impacting the entire OpenFlow network. To eliminate such threat, PermOF was proposed in Wen et al. (2013) which allows a minimum privilege to the applications. PermOF is a fine-grained permission mechanism that incorporates a customized permission set and a thread-based isolation system. In the same way, FortNOX was proposed in Porras et al. (2012), empowers OpenFlow security applications with the ability to produce enforceable flow constraints. It is a software extension that provides role-based authorization, rule reduction, conflict evaluation, policy synchronization, and security directive translation for the NOX OpenFlow controller. The authors in Shin and Gu (2013), show a new fingerprinting attack against SDN networks and further launch efficient resource consumption attacks. So, SDN brings new security issues that may not be ignored. A brief overview of the vulnerabilities present in the OpenFlow protocol as it is currently deployed by hardware and software vendors was provided in Benton et al. (2013). The authors also and also highlight the classes of vulnerabilities that emerge from the separation of the control plane and data planes in OpenFlow network designs. Furthermore, the security and dependability of the SDN has largely been a neglected topic and remains an open issue. The authors in Akhunzada et al. (2015) present a broad overview of the security implications of each SDN layer/interface.

4.2.6. Placement and latency

The position of the controller in the SDN architectures may impact the performance, reliability and scalability of an SDN. The authors in Heller et al. (2012), open the investigation by focusing on two specific questions: given a topology, how many controllers are needed, and where should they go? To answer these questions, we examine fundamental limits to control plane propagation latency. They show that the answers rely on the topology and that one controller location is often sufficient to meet existing reaction-time requirements. Similarly, the impact of the latency between an OpenFlow switch and its controller was discussed in Phemius and Bouet (2013). In this work, the authors show bandwidth arbitrates how many flows the controller can process, as well as the loss rate if the system is under heavy load, while latency drives the overall behavior of the network. The issue of placing controllers in SDN to maximize the reliability of control networks was handled in Hu et al. (2014b).The authors in this work, present a metric to

Table 3
Current controller implementations.

Controller	Implementation	Developer	Overview
NOX (Gude et al., 2008)	C++/Python	Nicira	NOX is the first Open Flow controller. NOX Classic was written in C++ and Python and current NOX is written in C++.
POX (Mccauley, 2016)	Python	Nicira	POX is a general, open-source SDN controller that supports OpenFlow controller.
SNAC (SNAC, 2016)	C++	Nicira	SNAC is an OpenFlow controller based on NOX-0.4. It supports a graphical user interface and a policy definition language and uses a web-based, user-friendly policy manager to manage the network, configure devices, and monitor events.
Maestro (Ng, 2010)	Java	Rice University	Maestro tries to improve the system throughput by exploiting multicore processors and parallelism. it provides interfaces for implementing modular network control applications and for them to access and modify network state.
Ryu (Ryu, 2016)	Python	NTT	Ryu is an SDN operating system that aims to provide logically centralized control and APIs to create new network management and control applications. It is a component-based SDN framework that supports OpenFlow v1.0, v1.2, and v1.3.
MUL (Mul, 2016)	C	Kucloud	MUL is an OpenFlow controller that supports a multi-threaded infrastructure and a multi-level northbound interface. It supports OpenFlow v1.0 and v1.3.
Beacon (Home - Beacon - Confluence, 2016)	Java	Stanford University	Beacon is a cross-platform and modular OpenFlow controller. It supports event-based and threaded operation.
Floodlight (floodlight/loxigen, 2016)	Java	BigSwitch	Floodlight supports a broad range of virtual and physical OpenFlow switches and can handle mixed OpenFlow and non-OpenFlow networks. It based on the Beacon implementation.
IRIS (IRIS, 2016)	Java	ETRI	IRIS is a recursive OpenFlow controller that aims to support scalability, high availability, and multi-domain support.
OESS (GlobalNOC, 2016)	Perl	NDDI	OESS is a set of softwares to configure and control dynamic VLAN networks using OpenFlow-enabled switches.
Jaxon (jaxon, 2016)	Java	Independent developer	Jaxon is a NOX-dependent OpenFlow controller.
NodeFlow (NodeFlow, 2016)	JavaScript	Independent developer	NodeFlow is an OpenFlow controller written for Node.JS.
ovs-controller (Open vSwitch, 2016)	C	Independent developer	ovs controller is a simple OpenFlow controller reference implementation with Open vSwitch for managing any number of remote switches through the OpenFlow protocol; as a result the switches function as L2 MAClearning switches or hubs (Open vSwitch, 2016)
Flowvisor (Sherwood et al., 2010)	C	ON.LAB	As a transparent proxy between OpenFlow switches and multiple OpenFlow controllers, Flowvisor allows multiple tenants to share the same physical infrastructure by dividing traffic flow space into slices.
RouteFlow (Nascimento et al., 2011)	C++	CPQD	RouteFlow provides virtualized IP routing services over OpenFlow-enabled hardware. RouteFlow is composed by an OpenFlow controller, an independent RouteFlow server, and a virtual network environment.
Helios (NEC Global, 2016)	C	NEC	Helios is an extensible OpenFlow controller that provides a programmable shell for performing integrated experiments (not publicly available).

characterize the reliability of SDN control networks and developed several placement algorithms.

4.3. Controller implementations

To date, different types of OpenFlow (compatible) controllers have been developed in the context of SDN which we will introduce in more detail in Table 3. All the controllers were reviewed here support the OpenFlow protocol version 1.0, unless stated otherwise.

4.4. Development tools

SDN simplifies network evolution and innovation by allowing rapid deployment of new services and protocols. In this section, we review currently available tools and environments for developing various aspects of SDN.

4.4.1. Simulators and frameworks

- *Mininet* (Team, 2016): It is an emulator platform using OpenFlow protocol, runs a set of end-hosts, switches, routers and links on a single Linux kernel by using lightweight virtualization. Components of Mininet act as real network components to check the possible bandwidth, the connectivity among nodes and deepest nodes, and the speed of flows.
- *Mininet-HiFi* (Handigol et al., 2012): Mininet-HiFi is an evolution of Mininet that enhances the container-based (lightweight) virtualization with mechanisms to enforce performance isolation, resource provisioning, and accurate monitoring for performance fidelity. One of the main goals of Mininet-HiFi is to improve the reproducibility of networking research.
- *Mininet CE* (Antonenko and Smelyanskiy, 2013) *and SDN Cloud DC* (Teixeira et al., 2013): They are extensions to Mininet for enabling large scale simulations.
- *NS-3* (ns-3, 2016): It supports OpenFlow switches within its environment, though the current version only implements OpenFlow v0.89.
- *OMNeT++* (Klein and Jarschel, 2013; Varga and Hornig, 2008): The OMNeT++ has been created with the simulation of communication networks, multiprocessors and other distributed systems. It supports OpenFlow v1.2 through a plugin.
- *EstiNet 8.0* (EstiNet Technologies, 2016): It supports many OpenFlow 1.3.2 and 1.0.0 switches. Besides this advantage, in the simulation mode of EstiNet; POX, NOX, Floodlight and Ryu controllers will have the role of SDN controller plane.
- *Trema* (Trema, 2016): Trema is a framework for developing OpenFlow controllers in Ruby and C.
- *Mirage* (MirageOS, 2015): Mirage is an Exokernel for constructing network applications across a variety of platforms and supports OpenFlow.

4.4.2. Debuggers

Programmable feature of SDN controller increases the probability of inadvertently errors; and generally, finding bugs is hard and time-consuming. So, debuggers are one of the important

components of OpenFlow/SDN. Debuggers are tools that test and diagnose program and provide programmers interact with program while it is executing on computer.

- *STS* (STS, 2016): SDN troubleshooting system is a simulator designed to allow developers to specify and apply a variety of test cases, while allowing them to interactively examine the state of the network.
- *NICE* (Canini et al., 2012): It presents efficient, systematic techniques for testing unmodified controller programs. NICE tool applies model checking to explore the state space of the entire system, the controller, the switches, and the hosts.
- *Cbench* (Oflops, 2016): Cbench is a program which tests controller performance by generating requests for packet forwarding rules and watching for responses from the controller.
- *OFLOPS* (Rotsos et al., 2012): It is a software framework which tests the capabilities and performance of OpenFlow-enabled software and hardware switches.
- *OFTest* (OFTestTutorial, 2016): OFTest is a Python-based framework that tests basic functionality of OpenFlow switches.
- *OFRewind* (Wundsam et al., 2011): OFRewind enables scalable, multi-granularity, temporally consistent recording and coordinated replay in a network, with finegrained, dynamic, centrally orchestrated control over both record and replay for troubleshooting problems in production networks.
- *VeriFlow* (Khurshid et al., 2012): It is a dynamic solution design and a layer between a SDN controller and network devices that analyzes and checks network configuration in real-time to find bugs without having negative impact on network performance.
- *FlowChecker* (Al-Shaer and Al-Haj, 2010): FlowChecker is a property-based verifier solution to identify any intra-switch misconfiguration within a single Flow table. The main information for debugging is gathered from flow table, different traffic statistics and controller messages.
- *HAS* (*header space analysis*) (Kazemian et al., 2012): HAS is a general and protocol agnostic framework, that allows statically checking of network specifications and configurations to identify an important class of network failures.
- *ATPG* (*automatic test packet generation*) (Zeng et al., 2012): ATPG is an automated and systematic approach for testing and debugging networks such as SDN. It generates a minimum set of test packets to (minimally) exercise every link in the network or (maximally) exercise every rule in the network to detect both functional (e.g., incorrect forwarding rules) and performance problems (e.g., a congested queue).
- *ndb* (Handigol et al., 2012): ndb is a prototype network debugger inspired by gdb, which implements two primitives useful for debugging an SDN: breakpoints and packet backtraces.

There also have been several works to troubleshoot bugs in SDN control software and improving OpenFlow development. The authors in Scott et al. (2012) propose the structure of the SDN software stack to automate the process of troubleshooting networks. They discuss two techniques for programmatically localizing the root cause of network problems: cross layer correspondence checking to find what problems exist in the network, and where in the control software the problem first developed; and simulation-based causal inference to identify when the triggering event(s) occurred.

4.4.3. Testbeds

Several testbeds have been built and deployed to allow multiple network experiments to be conducted concurrently in a production network. They support OpenFlow protocol.

- *PlanetLab Europe* (*PLE*) (OpenFlow support in PlanetLab, 2016): It supports OpenFlow capabilities through a sliver-ovs (modified version of OpenVSwtich). Experimenters can to create an OpenFlow overlay network by specifying the links between PLE nodes.
- *OpenFlow in Europe linking infrastructure and applications* (*OFELIA*) (Ofelia, 2016): OFELIA is a testbed in which researchers can dynamically control and extend the network via OpenFlow. OFELIA control framework (OCF) provides tools for user verification and access, allocation of the slice, configuration of the network.
- *Future internet testbed experimentation between brazil and Europe* (*FIBRE*) (Ciuffo, 2016): It aims to deploy and create a testbed based on OpenFlow which create common space between Brazil and Europe for future Internet architectures.
- *Internet2* (Internet2, 2016): Internet2 supports advanced network services, such as IPv6 and QoS and have provided wide area testbeds for the network research community, including support for projects such as PlanetLab, and GENI.
- *Research infrastructure for large-scale network experiments* (*RISE*) (JGN-X Website, 2016): This is another project that aims at building a large-scale international OpenFlow testbed based on JGN-X network in Japan.
- *SURFnet* (SURF, 2016): SURFnet is an OpenFlow testbed and it can be used for all forms of SDN-related work, such as gaining hands-on experience with OpenFlow switches and controllers, developing SDN prototypes and/or testing third-party SDN software.
- *California OpenFlow testbed network* (*COTN*) (The California OpenFlow Testbed Network, 2016): The OpenFlow-enabled COTN will aid researchers to the development of tomorrow's Internet using today's networks as a testbed for innovation. It connects with other OpenFlow testbed within the national research networks such as NLR and Internet2.
- *Emulab* (Schwarz et al., 2013): It is a network testbed that does not support OpenFlow, but provides network topologies defined by the user, in a controllable, predictable, and repeatable environment. So tries to add functionality to support OpenFlow.
- *Global environment for network innovations* (*GENI*) (GENI, 2016): GENI federated some platform such as PlanetLab, Internet 2, Emulab, etc. to support experimental research in networking by creating a huge testbed. It is supported by the National Science Foundation.
- *Open-access research testbed for next-generation wireless networks* (*ORBIT*) (Orbit, 2016): It is intended to be used to test and evaluate innovative protocols in real-world settings and it includes an OpenFlow-based network.

4.4.4. High level language

Some works were presented to assist SDN development by providing high-level abstractions; such as to translate application requirements into packet forwarding rules. So, this function dictates a communication protocol (e.g., a programming language) between the application plane and the control plane. SDN presents a simple, centralized programming mechanism for managing complex networks. However, there are some challenges in managing low-level details, such as installing and maintaining correct and efficient forwarding tables on distributed switches.

- *Maple* (Voellmy et al., 2013): Maple is a powerful mechanism that simplifies SDN programming by allowing a user to use a standard programming language to design an arbitrary, centralized algorithm, to decide the behaviors of an entire network for every packet entering. Additionally, to overcome the challenge of translating a high-level policy into sets of rules on distributed switches, it provides an abstraction that runs on every packet

entering a network.

- *Fault tolerating regular expressions* (*FatTire*)(Reitblatt et al., 2013): It is a new language for writing fault-tolerant network programs. FatTire is an example of a declarative language that based on regular expressions to allow users to v network paths with the degree of fault tolerance requirements.
- *Flow-based management language* (*FML*) (Hinrichs et al., 2009): FML is a high level declarative policy language that specifies management and security policies for OpenFlow networks. It, uses NOX to calculate and deploy flow table entries on switches.
- *Procera* (Voellmy et al., 2012): It is a language that applies the principles of functional reactive programming to provide a declarative, expressive, and compositional framework that allows operators to express network policies based on both reactive and temporal behaviors, which are typically necessary to express common, simple network policies.
- *Frenetic* (Foster et al., 2011; Foster et al., 2010): Frenetic language allows the programs written for one platform to work in other platforms. It eliminates complicated asynchronous and event-driven interactions between SDN applications and switching devices. Additionally, supports for designing a compiler, run-time environment, and modular programming constructs for SDN (Hinrichs et al., 2009), also constructs for certain tasks such as updates (Katta et al., 2013).

4.4.5. Industry standardizations, work/research group and forums

For developing, standardizing, coordinating, and implementing SDN and related technologies such as NFV (Network Function Virtualization) and cloud computing have been several approaches.

- *Open networking foundation* (*ONF*) (ONF Overview, 2016): ONF is the group that is most associated with the development and standardization of SDN. Particularly, they focus on standardizing the northbound and southbound interfaces.
- *Internet engineering task force* (*IETF*) (Internet Engineering Task Force (IETF), 2016): The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. The IETF Overlay Routing Area (NVO3, L2VPN, TRILL, LISP, PWE3), API (NETCONF, ALTO, CDNI, XMPP, SDNP, I2AEX), Controller (PCE, ForCES), Protocol (IDR, IS-IS, OSPF, MPLS, CCAMP, BFD), and Interface to the Routing System (I2RS) (Hares et al., 2013) working groups are also involved with SDN-related activities. For example, the ForCES (Tsou et al., 2012) working group defines protocols and interfaces for the separation of IP control and forwarding, centralized network control, and the abstraction of network infrastructure. Internet research task force (IRTF) (Active IRTF research groups, 2016) initiated an SDN working group to contribute to the community. Software-defined networking research group (SDNRG) (IRTF Software-Defined Networking Research Group (SDNRG), 2016) discusses SDN from various perspectives (e.g., scalability, abstractions, and programming languages) to identify approaches that can be used in near-term and future research challenges.
- *International telecommunication union telecommunication standardization sector* (*ITU-T*) (ITU Telecommunicatio, 2016): ITU-T is capable of undertaking comprehensive discussions on the public networks, particularly with regard to their international connections, of covering billing issues and of aspects related to restrictions. Some ITU-T study groups are working on requirements for SDN. For example, ITU-T SG13 is developing SDN-related questions and network virtualization. ITU-T SG11 is investigating signaling requirements and protocols for SDN, and this work aligns with the functional requirements and architectures developed by ITU-T SG13. Joint coordination activity on SDN (JCA-SDN) (Joint Coordination Activity on Software-Defined Networking (JCA-SDN), 2016) is established for coordinating and helping plan work to ensure that the ITU-T SDN standardization progresses in a well-coordinated manner among relevant SGs (e.g., SG13 on use-cases, requirements, and architecture; SG11 on protocols and interoperability).
- *Metro Ethernet forum (MEF)* (MEF, 2016): MEF is a nonprofit international industry consortium and now introducing SDN technologies to the Carrier Ethernet paradigm.
- *European telecommunications standards institute* (*ETSI*) (ETSI - European Telecommunications Standards Institute, 2016): ETSI is an independent, not-for-profit, standardization organization in the telecommunications industry (equipment makers and network operators) in Europe, and owns an initiative on NFV.
- *Institute of electrical and electronics engineers* (*IEEE*) (IEEE, 2016): Its objectives are the educational and technical advancement of electrical and electronic engineering, telecommunications, computer engineering and allied disciplines. The IEEE SDN (Home, 2016) Initiative is composed by seven committees: Conference, Education, Publications, Publicity, Standards, Pre-industrial and Outreach. Although IEEE is still not involved deeply with SDN, we can find some 802.1 Overlay Networking projects that look at SDN-related concepts.

The open source software community, including OpenDaylight (The OpenDaylight Platform, 2016), OpenStack (Home, 2016), and Apache CloudStack (Apache Cloudstack, 2016) is developing basic building blocks for SDN implementation for the advancement of SDN. For example, OpenDaylight is intended to be extensible and configurable to potentially support emerging SDN open standards (e.g., OpenFlow).

Table 4 summarizes existing efforts for controller design aspects, development tools and industry standardizations.

5. Application layer

The application layer is at the top of the SDN architecture, which includes all the applications that exploit the services provided by the controller in order to perform network-related tasks in various areas.

5.1. Traffic engineering

A dynamic and efficient network management needs information about current network status and timely change of network control. The authors in (Handigol et al., 2010) present Aster*x, a prototype distributed load balancer, which uses OpenFlow to measure the state of the network and makes load balancing scalable, dynamic, and flexible to directly control the forwarding paths. Similarly, in (Agarwal et al., 2013) the authors

Table 4
Summarizes existing efforts for the control plane.

Name	Description
Controller design aspects	State consistency, scalability, flexibility, availability, security, placement and latency
Simulation and Framework	Mininet, Mininet-HiFi, Mininet CE, SDN Cloud DC, ns-3, OMNeT++, Trema EstiNet 8.0, Mirage,
Debuggers	STS, NICE, Cbench, OFLOPS, OFTest, OFRewind, Frenetic, VeriFlow, FlowChecker, HAS, ATPG, ndb
Testbeds	PlanetLab Europe, OFELIA, FIBRE, Internet2, RISE, SURFnet, COTN, Emulab, GENI
Programming Languages	Maple, FatTire, FML, Procera, Frenetic
Standards	ONF, IETF, IRTF, SDNRG, ITU-T, MEF, ETSI, IEEE

utilize the network information (collected by the centralized SDN controller) to get significant improvements in network utilization and reduce packet losses and delays. Particularly, they show these improvements are possible even in cases where there is only a partial deployment of SDN capability in a network. Also, they formulate the SDN controller's optimization problem for traffic engineering with partial deployment. Networks today rely on middle-boxes to provide critical performance, security, and policy compliance capabilities. Plug-n-Serve (Handigol et al., 2009) is an Open-Flow based server load-balancing solution that leverages an OpenFlow controller to effectively reduces response time of web services in unstructured networks built with cheap commodity hardware. SIMPLE (Qazi et al., 2013) is an SDN-based policy enforcement layer for efficient middle-box-specific traffic engineering. SIMPLE exploits SDN technologies to ensure that the traffic is directed through the desired sequence of middle-boxes to realize efficient middle-box applications. In the work (Macapuna and Rothenberg, 2010) a novel data center architecture based on load-balanced forwarding with in-packet Bloom filters enabled proposed, by two support services that distribute the directory and topology state of OpenFlow controller applications. A general framework to formulate the multi-flow update sequence scheduling problem and come up with an optimal flow migration ordering solution was proposed in (Liu et al., 2015), which minimizes the maximum link utilization and significantly reduces the possibility of congestion.

Recently, some researchers proposed enhancement methods for QoS over SDN. The authors in (Civanlar et al., 2010) proposed an architecture to support QoS flows in an OpenFlow environment with a centralized controller and many forwarders. They focused on the analytical framework for optimization of the QoS flow routing, and the functionality needed within the controller and forwarders to efficiently support QoS and also describe the control layer messaging between the controller and forwarder to set up queues, detect congestion and reroute traffic streams that require QoS. OpenQoS (Egilmez et al., 2012), is a novel OpenFlow controller design for multimedia delivery with end-to-end QoS support. This approach is based on QoS routing where the routes of multimedia traffic are optimized dynamically to fulfill the required QoS. OpenQoS can guarantee seamless video delivery with little or no video artifacts experienced by the end-users. A network QoS control framework was proposed in (Kim et al., 2010) for automated fine-grained management of converged network fabric. The QoS controller can create network slices to assign different applications traffic to different slices, and provision the slices dynamically to satisfy the performance requirements across all applications. Video over software-defined networking (VSDN) (Owens and Durresi, 2013), is capable of selecting the optimum path for video applications which can improve the QoS of video applications. A VSDN API allows application developers to request service from VSDN enabled networks.

5.2. Network management

There have been some works that propose new network management systems using SDN technologies. Integrated network management and control system (I-NMCS) framework (Sharma et al., 2013) combines legacy network management functions (e.g., discovery, fault detection) with the end-to-end flow provisioning and control enabled by SDN. In Kim and Feamster (2013), to simplify and improve various aspects of network operations and management, such as enabling frequent changes to network conditions and state, providing support for network configuration in a high-level language, and better visibility and control over tasks such as network diagnosis and troubleshooting, the authors proposed Procera. Procera is an event-driven network control framework based on SDN. The authors of Bennesby et al. (2012) propose how SDN provide an inter-domain routing component to an SDN control plane. They implement it using an NOX-OpenFlow architecture that was originally created only for routing in enterprise networks. VeriFlow (Khurshid et al., 2012) is a verification tool in order to achieve a real-time checking in SDN networks to keep OpenFlow rules consistent and thus control the network traffic in a stable way.

5.3. Measurement and monitoring

FleXam (Shirali-Shahreza and Ganjali, 2013) is a flexible sampling extension for OpenFlow controller to access packet-level information, In other words, the controller can define which packets should be sampled, what part of packet should be selected, and where they should be sent. In Takagiwa et al. (2013), the authors proposed SoR-based programmable network for effective future SDN which enables layer-2 to layer-7 information to be captured, analyzed, and stored and has a high-throughput database (DB) and is able to analyze all transactions on its interfaces. Also, SoRs can provide APIs to access stored contents in order to enrich services. Similarly, Atlas (Qazi et al., 2013) is a framework which incorporates application awareness into SDN, which is currently capable of L2/3/4-based policy enforcement but agnostic to higher layers. It enables fine-grained, accurate and scalable application classification in SDN. Another tool proposes a software-defined traffic measurement architecture called OpenSketch (Yu et al., 2013), it proposed to support more customized and dynamic measurement while guaranteeing the measurement accuracy. To achieve this goal, OpenSketch separates the measurement data plane (supporting many measurement tasks) from the control plane (configuring and allocating different measurement tasks).

5.4. Middle-box

Middle-boxes (for example a load balancer) and in-network services (for example in-network caching) are very popular in today's networks. SDN technologies can be used to simplify the network by eliminating it from middle-boxes and integrating their functionality within the network controller to produce better middle-box-based services. Dynamic and traffic-dependent modifications enforced by middle-boxes make it difficult to reason about the correctness of network-wide policy enforcement (e.g., access control, accounting, and performance diagnostics). FlowTags (Fayazbakhsh et al., 2013) is an extended SDN architecture in which middle-boxes add tags to outgoing packets so that tags are used on switches and (other) middle-boxes for systematic policy enforcement. In addition, SDN simplifies to embed various in-network services, such as advertisement targeting (Nishida and Nakao, 2012), where a software control in the middle of the network injects related advertisements based on the session content. In the similar approach, in Lee et al. (2011) an In-network processing (INP) framework was proposed which orchestrates various computing resources and network devices and enables seamless and efficient deployments of network services. CoMb (Sekar et al., 2012), is a new architecture for middle-box deployments that systematically explores opportunities for consolidation, both at the level of building individual middle-boxes and in managing a network of middle-boxes. The authors in Gember et al. (2012) try to realize a software-defined middle-box networking framework to simplify management of complex, diverse functionalities and engender rich deployments. They discuss the major challenges that arise—representing, manipulating, and knowledgeably controlling middle-box state—and also present initial thoughts on the appropriate abstractions and interfaces to address them.

5.5. Security and dependability

The centralized control of SDN is useful for implementing security applications. Supervision of SDN on whole network flow and monitoring behavior of users makes SDN possible to detect attack rapidly and prevent more damage. OpenFlow random host mutation (OFRHM) (Jafarian et al., 2012) uses OpenFlow to develop a moving target defense (MTD) architecture that transparently mutates IP addresses with high unpredictability and rate, while maintaining configuration integrity and minimizing operation overhead. In this approach, the OpenFlow controller dynamically allocates a random virtual IP (translated to/from the real IP of the host) to each host to avoid exposing an authentic IP that could be used by the attacker. The authors in Mehdi et al. (2011), have shown that SDN OpenFlow and NOX allow flexible, highly accurate, line rate detection of anomalies inside Home and SOHO (small office/home office) networks. The standardized interface provided by a SDN would allow our applications to be updated easily as new security threats emerge while maintaining portability across a broad and diverse range of networking hardware. Resonance (Kim et al., 2015) is a system for securing enterprise networks, where the network elements themselves enforce dynamic access control policies based on both flow-level information and real-time alerts. Resonance uses programmable switches to manipulate traffic at lower layers and allows the switches to take actions (e.g., dropping) to enforce high-level security policies and distributed monitoring and inference systems. Some works exploit SDN capabilities to develop applications such as edge-based authentication gateways (Suenaga et al., 2012; Braga et al., 2010).

5.6. Virtualization

Migrating virtual machines (VMs) along with their connections has numerous benefits in data centers, ranging from load balancing to power saving to optimization of performance and utilization by VM reallocation. However, directly migrating individual components can lead to inconsistencies and overloads of resources. In Ghorbani and Caesar (2012), the authors use the abilities of SDN (i.e., running algorithms in a logically centralized controller and manipulating the forwarding layer of switches) to handle VM migration issues. Particularly, given the network topology, service-level agreement (SLA) requirements, and set of VMs that are to be migrated, along with their new locations, the algorithms (running in the SDN controller to orchestrate these changes within the network) output an ordered sequence of VMs to migrate, also a set of forwarding state changes. LIME (live migration of ensembles) (Keller et al., 2012) is an SDN-based solution for live migration of VMs, which handles the network state during migration and automatically configures network devices at new locations. The traditional IP multicast technique based on IGMP (internet group management protocol) is not scalable because it consumes a large amount of resources such as IP multicast tables and CPU time. the authors in Nakagawa et al. (2012), extended OpenFlow to manage an IP multicast in overlay networks. In particular, they eliminate periodic Join/Leave messages and achieve more than 4,000 tenants. It is as well as possible to combine SDN and NFV to provide a virtual networking lab for computer science education without any simulation (Choi).

5.7. Networks

5.7.1. Wireless and mobility

SDN also can be deployed for wireless and mobility networks. Currently, main research focus in SDN wireless and mobility architecture is centralized control of them. Odin (Suresh et al., 2012), is a SDN framework to introduce programmability in enterprise wireless local area networks (WLANs) by exploiting a light virtual AP abstraction. Enterprise WLANs need to support a wide range of services and functionalities (e.g., authentication, authorization and accounting, mobility, interference management, load balancing and AP association decision). Odin allows a network operator to implement enterprise WLAN services as network applications while not requiring client-side modifications. OpenRadio (Bansal et al., 2012) is a novel design for a programmable wireless data plane that provides modular and declarative programming interfaces across the entire wireless stack. OpenRadio can be used to realize modern wireless protocols, such as Wi-Fi and LTE, while providing flexibility to modify the PHY and MAC layers. OpenRoads (Yap et al., 2010) is an open-source platform for innovation in mobile networks over NOX. OpenRoads or in other words, OpenFlow Wireless aims to create an open platform where various mobility solutions, network controllers, and routing protocols are examined. OpenRoads provides flexible control of the data path and device configuration using OpenFlow and SNMP, respectively. This allows researchers to implement wildly different algorithms and run them concurrently in one network. The authors in Yap et al. (2010) propose an open (but backward compatible) wireless network infrastructure that can be easily deployed on college campuses worldwide. An architecture that integrates OpenFlow with WMNs (wireless mesh network) was proposed in Dely et al. (2011), which provides flow-based routing and forwarding capabilities. SoftRAN (Gudipati et al., 2013), is a centralized software-defined radio access network designed for performing handovers efficiently. In SoftRAN all the base stations are abstracted and controlled in a centralized way. The centralized control of SDN can also be used to achieve complete virtualization and programmability in radio access networks (RAN). OpenRAN (Yang et al., 2013) is an architecture for software-defined RAN via virtualization. It provides open, controllable, flexible and evolvable wireless networks. For handling significant scalability issues exist in cellular networks and enable new services, Li et al. (2012) proposed a cellular SDN architecture with local control agents with ability to make simple decisions. The centralized controller responsible for interpreting flows with high level abstractions. SDN can be applied in wireless sensor networks (WSNs) (Luo et al., 2012). Generally, using SDN in WSNs provided the SDN benefits such as flexibility, easier management, optimized resource utilization, congestion control (Ghaffari, 2015) etc.. The network controllers have the power to set policies to support several applications by utilizing sensor based software-defined wireless network. Also this approach would permit using the same sensor nodes for several applications. In (Wang et al., 2016) a SDN based Sleep Scheduling algorithm SDN-ECCKN is proposed to manage the energy of the network. In Guimaraes et al. (2013), the authors explore SDN mechanisms, which exploit OpenFlow to optimize handovers in heterogeneous wireless environments, particularly for media-independent handover procedures. To apply concepts of abstraction to wireless ad hoc network of smartphone, SDN in ad hoc networks (Baskett et al., 2013) was developed. This Hybrid platform has been implemented on Android operating system. The purposed platform is more modular and easier for modification and extension of its components. The authors in Trivisonno et al. (2015), proposed a novel plastic architecture for the advanced 5 G network infrastructure by harvesting latest advances of SDN, network functions virtualization and edge computing. Other use case were mentioned in Brief (2013) referred the benefit of based SDN. SoftCell (Jin et al., 2013) is scalable architecture that supports fine-grained policies for mobile devices in cellular core networks, using commodity switches and servers was proposed in. SoftCell enables operators to realize high-level service policies that direct traffic through sequences of middle-boxes based on subscriber attributes and applications. WiVi (Nakao, 2016) is a Wi-Fi network

virtualization infrastructure that not only enables multiple coexisting access points to work concurrently, but also enables data plane programmability for potential application developers. Two latest and promising innovations of Internet, SDN and network virtualization, to mobile and wireless scenarios was discussed in Yang et al. (2014).

5.7.2. Optical network

There is a great deal of benefits when adopting SDN/OpenFlow for optical network control (Open Networking Foundation, 2016). SDN provides many benefits such as improved network control, programmable, abstracted interface for flexible application reconfigurations in optical control units. The potential benefits and challenges of extending SDN concepts to various transport network architectures include optical wavelength and fiber switches, circuit switches, and sub-wavelength optical burst switches are discussed in Gringeri et al. (2013). The combination of SDN controller with optical switching to explore the tight integration of application and network control discussed in Wang et al. (2012). In addition it particularly studies the run-time network configuration for big data applications to jointly optimize application performance and network utilization. It shows that the combination has great potential to improve application performance with relatively small configuration overhead. The authors in Shirazipour et al. (2012), argued that the applying SDN to circuit based transport networks could be the enabler for both packet-optical integration and improved transport network applications. They discuss extensions to OpenFlow v1.1 to achieve control of switches in the multi-technology transport layer. Moreover, a unified OpenFlow/GMPLS (generalized multiprotocol label switching) control plane that can be used to provide GMPLS-specific features to OpenFlow networks was proposed in Azodolmolky et al. (2011). Similarly, OpenFlow-based control plane in Flexi-Grid optical networks (Zhang et al., 2013), feasible the dynamic light path establishment and adjustment via extended OpenFlow protocol. In Patel et al. (2013), the authors introduce a Software-defined optical network (SDON) architecture and develop a QoS-aware unified control protocol for optical burst switching in OpenFlow-based SDONs. OpenFlow is also exploited to dynamically create a bidirectional wavelength circuit for a TCP flow (Gudla et al., 2010) or wavelength path control for light-path provisioning in transparent optical networks (Liu et al., 2011). A simple programmable architecture in Sadasivarao et al. (2013) was proposed which abstracts a core transport node into a programmable virtual switch, that meshes well with the software-defined network paradigm while leveraging the OpenFlow protocol for control.

5.7.3. Home and small networks

Small networks such as those found in the home or small businesses have become increasingly complex and prevalent with the widespread availability of low-cost network devices, the need for more careful network management and tighter security has correspondingly increased. Several projects have examined how SDN could be used in them. The authors in Mortier et al. (2012), believe that users desire greater understanding and control over their networks' behavior; and present a prototype for a home network in which SDN is used to provide users a view into how their network is being utilized while offering a single point of control. In managing and troubleshooting home networks, one of the challenges is in knowing what is actually happening. So, the authors in Calvert et al. (2011) proposed instrumenting the network gateway/controller to act as a Home Network Data Recorder to create logs that may be utilized for troubleshooting or other purposes. By exploiting the SDN switches that enable flexible remote management, the authors in Calvert et al. (2011) propose an architecture for home network security which outsources the management and operations of these networks to a third party (e.g. ISP) which has a broader view of network activity. The controller applies distributed inference to detect performance and security problems in the local networks.

5.7.4. Cloud and data center networking

OpenFlow enables a network to reduce energy consumption by selectively powering down links and redirecting traffic to alternate paths during periods of lighter load. One approach to applying OpenFlow to energy savings in the data center, called ElasticTree (Heller et al., 2010). ElasticTree is a network-wide energy optimizer that monitors data center traffic conditions and chooses the set of network elements that must stay active to meet the performance requirements. The authors discuss several strategies to find minimum-power network subsets and show energy savings between 25% and 62% under varying traffic conditions. The authors in Kannan and Banerjee (2012), exploit the capabilities of SDN and proposed Scissor which tries to save energy by removing redundant traffic. Scissor effectively replaces the redundant header information with a flow ID to be used for the forwarding. Scissor leverages SDN technologies to dynamically allow switches to route packets based on the flow IDs. NCP (Mann et al., 2013), is a system that uses network based replication to enable service replication in data centers through software-defined networking. NCP allows its users to identify flows based on network addresses and ports and to specify a replication target for each such flow. NCP identifies flows based on network addresses and ports, and specifies a replication target for each identified flow. NCP then determines the ideal switch for replication and installs corresponding forwarding rules so that the identified flow goes to a designated server. Some work tries to satisfy the need for customized routing and management of distributed data centers that cannot be easily achieved by a traditional WAN (wide area network) architecture. B4 (Jain et al., 2013) designed by Google, is a hybrid approach with simultaneous support of existing routing protocols and novel OpenFlow SDN approach. The centralized traffic engineering is applied to easily satisfy massive bandwidth requirements, maximize average bandwidth, and enable rate limiting and measurement at the edge. They address the critical performance and reliability issues that WANs faced when delivering terabits per second of aggregate bandwidth across thousands of individual links. By exploiting the global network view enabled by the SDN paradigm, SWAN (Hong et al., 2013) is a software-driven WAN (SWAN) proposed by Microsoft, utilizes policy rules to allow inter-data center WANs to carry significantly more traffic for higher-priority services, while maintaining fairness among similar services. SWAN controls when and how much traffic each service sends, and re-configures the data plane to match current traffic demand. M2cloud (Liu et al., 2013) is a software-defined framework providing scalable network control for multi-site data centers. M2cloud employs two-level controllers with decoupled functions, providing each tenant with flexible virtualization support in both intra- and inter-data center networks. In addition, SDN provides opportunities to extend the service provisioning model of infrastructure as a service (IaaS). CloudNaaS (Benson et al., 2011), is a network service platform that enables tenants to leverage many of the network functions needed for production enterprise applications to run in IaaS clouds.

5.7.5. Information-centric networking

Recently, many researchers claimed that current internet architecture is not able to response the emerging and future need of users. There for, new architectures were introduced such as information centric network (ICN) to solve fundamental limitations. ICN is a new paradigm proposed for the future architecture of the Internet, which supports content-oriented services (Ahlgren et al.,

2012). ICN is a novel networking paradigm which promises to provide technological solutions that best fit with the way in which Internet is actually utilized. Assessment of proposed solutions requires appropriate experimental testbeds. In this regard, SDN is a valuable tool to build a testbed for ICN (Melazzi et al., 2012; Salsano et al., 2013) and also SDN features can also be utilized to realize ICN capabilities in an efficient manner. In Veltri et al. (2012), the authors have discussed some issues related to the application of SDN concepts to ICN and also, how ICN functionalities can be implemented over an OpenFlow network and how OpenFlow should be modified to better suit ICN functionalities. They present which how SDN and ICN could concretely be combined, deployed, and tested. In addition, they implemented a possible realization of a novel design for ICN solutions and point to possible testbed deployments for future testing. Caching strategy might dramatically influence performance and efficiency of content-centric networks. In Nguyen et al. (2013), an OpenFlow-based architecture that performs efficient caching for content-centric networks was proposed. C-flow (Suh et al., 2012) seeks to achieve efficient content delivery by leveraging the current OpenFlow functionalities. C-flow can deliver content to mobile hosts by using the byte-range option of the HTTP header. Multicasting/unicasting is naturally supported by mapping between files and their corresponding IP addresses. A routing protocol (Torres et al., 2012) which supports mobility by means of controller, can be easily implemented using software-defined ICN. The authors in (Ren et al.), discussed the role of virtualization in NDN (Named Data Networking), an architecture based on ICN, and outlined traffic optimization, traffic engineering and in-network catching management as the advantages of implementing NDN over SDN, specially for multimedia traffics. In Ooka et al. (2013), the authors present the detailed design and implementation of an OpenFlow-based CCN (content-centric networking) with the primary aim to achieve forwarding and end-to-end communication.

6. Research challenges and future direction

The overview of the related studies in this paper indicates that both industrial and academic settings have become highly interested in SDN and its protocols such as OpenFlow. The solutions provided by SDN have their own challenges and lead to new research questions which should be addressed in the future works. Some of the most generic questions are: how to optimize SDN? How to apply SDN in different networks? How to establish a tradeoff between different SDN implementations? In this section, different challenges which need be investigated in the upcoming studies are mentioned as directions for further research.

6.1. Data plane

6.1.1. Data plane programmability

Research into SDN focuses mainly on the development and programmability of the control plane. Although supporting data plane programmability is feasible, very few studies have addressed this issue and it is regarded as an under-researched issue in most SDN-related studies (Farhad et al., 2014). Research should be conducted on all three planes of SDN so as to better figure out programmability in networking stack. Indeed, many recent schemes such as SDN, NFV and network service chaining (NSC) have been proposed so that networks can be converted into programmable platforms with a focus on control plane (SDN) and data plane, respectively (NFV/NSC). Fig. 4 illustrates DPN (deeply programmable network) which is regarded as the overall solution for programming both data and control planes deeply (Fig. 5).

SDN can be extended to provide simple programmability for data plane and support interface definition or redefinition for it which reduces maintenance complexity and decreases life-cycle costs related to hardware-based inflexible data plane elements. FLARE (Nakao, 2012) can be considered as one of the current efforts for accomplishing DPN.

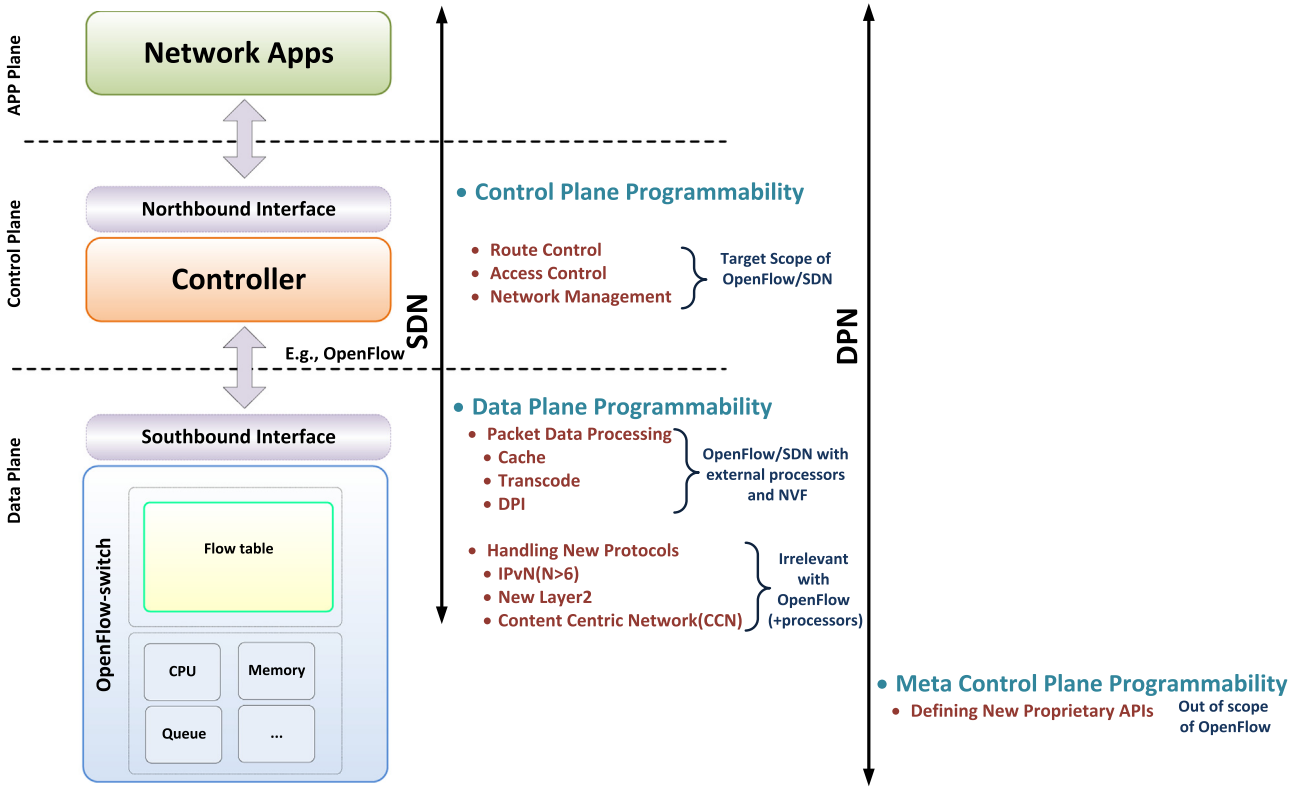

Fig. 5. Deep programmability within network.

6.1.2. Southbound APIs

Although OpenFlow significantly helps customize control plane of a router, at the present, there are no available solutions for data plane customization. Indeed, APIs are required to achieve different tasks for data plane similar to OpenFlow for control plane. Inasmuch as data plane can look inside a packet, hence, many applications such as deep packet inspection can be implemented and used. It should be noted that the major challenge and difficulty is to make a balance between efficiency and flexibility of data plane APIs. That is to say, enough throughputs should be maintained as well as packet processing mechanisms should be deployed as the data plane APIs should deal with a delay-sensitive environment. The first implementation of edge router was introduced in (Risso and Cerrato, 2012) which makes it possible to customize data plane processing.

6.1.3. Extensibility and platform independency

Both common hardware-centric networking and OpenFlow-compatible switches make networking dependent on a special family of hardware or software which impedes innovation and extensibility. The independence of networking from particular hardware or protocols can result in more productivity and innovation. This limitation and dependency can motivate researchers and experts to enhance SDN independency from all kinds of underlying technologies. In line with this purpose, commodity (e.g., x86 architecture) or programmable hardware (e.g., NPU and GPU) can be regarded as remarkable movements towards SDN independence. In a similar vein, regarding software-related issues, SDN should be developed in such a way that it its dependencies are minimized. As discussed in (Song, 2013), POF (protocol-oblivious forwarding) can eliminate such dependencies of protocol-specific configurations on forwarding elements. Also, it can enhance data-path with new statefull instructions or actions to support genuine SDN behavior.

6.1.4. Switch designs

Recent OpenFlow switches are diverse and differ remarkably from one another in terms of feature set (flow table size, optional actions), performance (fast vs. slow path, control channel latency/throughput), interpretation and adherence to the protocol specification and architecture (hardware vs. software designs).

6.1.4.1. Heterogeneous implementation

The type of implementation has a fundamental impact on behavior, accuracy, and performance of switches including differences from flow counter behavior to other performance metrics. One method to accommodate such heterogeneity is through NOSIX, a portable API that separates the application expectations from the switch heterogeneity (Yu et al., 2014).

6.1.4.2. Flow table capacity

Flow matching rules are stored in flow tables within network devices. One practical difficulty is related to providing switches with large and efficient flow tables so that rules can be stored (Appelman and de Boer, 2012). TCAMs (Ternary content addressable memories) are a common choice for holding flow tables. Although TCAMs are flexible and efficient with respect to matching capabilities, they are costly and typically small (from 4 K to 32 K entries). Some research studies have focused on compression techniques for reducing the number of flow entries in TCAMs (Braun and Menth, 2014; Agarwal et al., 2014).

6.1.4.3. Performance

throughput of commercial OpenFlow switches can vary from 38 to 1000 flow-mod per second where most devices achieve throughput values less than 500 flow-mod per second. This issue is clearly considered to be a limiting factor which should be taken into account in the switch design process. As the procedure proposed in Mogul and Congdon (2012), one method for handling this problem is to add more powerful CPUs into switches.

6.1.4.4. Evolving switch designs and hardware enhancements

As it can be observed in software/hardware innovation cycle, certain hardware improvements are required to optimize SDN capabilities and performance. Novel SDN switch designs are appearing where a myriad of hardware combinations such as SRAM, RLDRAM, DRAM, GPU, FPGA, NPs and CPUs work together with TCAMs among other specialized network processors. Similar to the method proposed by the Rain Man firmware (Stephens, 2012), alternatives to TCAM-based designs include new hardware architectures and components in addition to new and more scalable forwarding planes.

6.1.4.5. Native SDN switch designs

The majority of studies for (re)designing SDN switches have usually followed evolutionary approaches for retrofitting specific programmable features into existing hardware layouts. Such studies are based on common wisdom on switch/router designs and consolidated technologies such as SRAM, TCAM, FPGA. One deviation from this approach is the ongoing research study on forwarding metamorphosis (Bosshart et al., 2013) which is a reconfigurable match table model inspired from RISC-like pipeline architecture applied to switching chips.

6.2. Controller platforms

As mentioned above, controller platform is deemed to be a significant component of SDN architecture. Hence, studies should be conducted to enhance the following factors in SDN controllers: performance, scalability, distribution, modularity, highly available programmer-friendly software. As a case in point, distributed controller platforms should deal with a number of challenges. That is, the latency between forwarding devices and controller instances, fault-tolerance, load-balance, consistency and synchronization can be regarded as some of the significant challenges. Operators should consider and figure out how the combination of different functions and modules can improve the network.

6.2.1. Modularity and flexibility

A number of ongoing research studies are aimed at the modular and flexible composition of controllers. For instance, RAON (Park et al., 2014) is regarded as a recursive abstraction of OpenFlow controllers where each controller can observe the controllers below OpenFlow switches. Hence, research gaps in this area include the definition of suitable interfaces between different layers in a hierarchy of controllers.

Interoperability and application portability: Like forwarding devices, vendor agnosticism which stems from standard southbound interfaces is critical for fostering interoperability between controllers. Early studies favoring more interoperable control platforms include portable programming languages such as Pyretic (Monsanto et al., 2013) and east/westbound interfaces among controllers such as SDNi (Yin et al., 2012). Nevertheless, these efforts are yet far from fully realizing controller interoperability and application portability.

High-Availability: with respect to production, SDN controllers should maintain their proper functioning even under the pressure of different objectives from the applications they host. Indeed, many improvements and optimizations are required so that potential risk vectors of controller-based solutions can be handled (Kreutz et al., 2013). Future studies should introduce consistent, fault-tolerant data stores for building reliable distributed controllers (Botelho et al., 2013, 2014; Berde et al., 2014).

Delegation of control: for enhancing operational efficiency, SDN controllers can delegate control functions for reporting state and attribute value changes, threshold crossing alerts, hardware failures, etc. These notifications typically follow a publish/subscribe model, i.e., controllers and applications subscribe (on-demand) to the particular class of notifications they are interested in. Furthermore, these subsystems can provide resilience and trustworthiness properties (Kreutz et al., 2012).

6.3. User-driven control

The majority of SDN APIs were designed to be used by network operators and/or administrators. Although these APIs are useful, some APIs should be also implemented by users (Ferguson et al., 2013). For end-users, APIs can be utilized to realize on-demand services. For instance, an intrusion detection application on a user machine can request network to manipulate traffic from a specific source. On the other hand, a MapReduce-style application can ask for bandwidth guarantees for optimizing performance of its shuffle phase. Such instances can be interpreted in a way that API should be present between network control plane and its client applications. These rules require read and write access so as to figure out network status and make independent configuration variations for their own benefit, respectively (Ferguson et al., 2013). With respect to the purpose of user-defined controlling, many challenges and difficulties should be sorted out. As a case in point, trust may play an important role in such APIs since a section of network control should be assigned to a semi-external entity. Moreover, the conflicts among various users' requests should be eliminated and, meanwhile, baseline fairness and security should be maintained.

6.4. Resilience

Establishing resilient communication is deemed to be one of the critical objectives in networking. Hence, SDNs should be able to yield the same degrees of availability as legacy and other modern technological alternatives. One important research question regarding split control architectures like SDN is Desai and Nandagopal (2010) related to their actual resilience against faults which might compromise communications ween control and data planes. Consequently, it leads to the production of “brainless” networks. Indeed, the malfunctioning of particular SDN elements should not destroy availability. When critical control plane functions such as those related to link failure detection or fast reaction decisions are considered, the relocation of SDN control plane functionality from inside the boxes to remote, logically centralized locations is a challenge. OpenFlow network resilience is a function of both fault-tolerance in data plane (as in traditional networks) and high availability of the (logically) centralized control plane functions. Thus, it can be argued that SDN resilience is a thorny issue due to the multiple possible failures of the different pieces of the architecture. As discussed in Kim et al. (2012), there is a notable research lacuna with regard to building and operating fault-tolerant SDNs. Indeed, Google B4 (Jain et al., 2013) can be considered as one of the few cases which indicated that SDN can be resilient. Distributed controller architectures are instances of approaches towards resilient SDN controller platforms with different compromises regarding consistency, durability and scalability.

6.5. Performance evaluation

Numerous implementations of OpenFlow from hardware and software vendors are used in different networks from small enterprises to large-scale data centers. Hence, an increasing number of experiments are expected to be done on SDN-enabled networks in the near future. Hence, future studies will inevitably raise more research questions and gaps and these questions on SDN performance and scalability should be systematically investigated. The overview of the related works in this study revealed that few studies have evaluated the performance of OpenFlow and SDN architecture. Although simulation studies and experimentation are among the most widely used performance evaluation techniques, analytical modeling has its own benefits too. A closed-form description of a networking architecture paves the way for network designers to have a quick (and approximate) estimate of the performance of their design, without the need to spend considerable time for simulation studies or expensive experimental setup. Some work has investigated ways to improve the performance of switching capabilities in SDN. These mainly consist of observing the performance of OpenFlow-enabled networks regarding different aspects, such as lookup performance (Jarschel et al., 2011), hardware acceleration (Luo et al., 2009), the influence of types of rules and packet sizes (Bianco et al., 2010), performance bottlenecks of current OpenFlow implementations (Curtis et al., 2011), how reactive settings impact the performance on data center networks (Pries et al., 2012), and the impact of configuration on OpenFlow switches (Oflops, 2016).

6.6. Deployment

In the past, SDN was mainly applied in networks of academic settings and data centers. Nevertheless, in more recent studies, SDN has been expanded to an extensive range of networks from optical, home, wireless, cellular networks to ICN (Section 5.7). Inasmuch as each network has specific settings, the application of SDN to new networks has created opportunities and challenges which should be addressed in future studies (Feamster et al., 2014). One more research issue is related to incremental deployment. Many of the studies using SDN suppose a complete SDN deployment. However, in real-life situations, one part of network can only be updated at a time in case budgets are limited. Hence, one problem is that the compatibility between existing network components and SDN-enabled components should be supported (e.g., (Lin et al., 2013)). Another challenge is related to specifying which existing switches or routers should be upgraded for enhancing SDN advantages (Levin et al., 2013).

Furthermore, inter-networking across multiple SDN domains should be considered as a research lacuna. The majority of recent studies on SDN have investigated the context of a single administrative domain. One important research question is related to logically centralized SDN control. Nevertheless, it should be noted that logically centralized control might not be appropriate for multiple SDN networks since in these networks, controls are independently driven by their own controllers. As a case in point, physically disseminated SDN networks (including testbeds) require agreement from the corresponding administrators. moreover, controller might have to expanded so as to cover inter-networking.

6.7. Virtualization and cloud services

SDN paradigm can be used in carrier networks as a technological tool for sorting out some generic and old issues. The

followings are regarded as new architectures for a smooth migration from:

- Current mobile core infrastructure to SDN (Pentikousis et al., 2013), and techno-economic models for virtualization of these networks (Naudts et al., 2012).
- Carrier-grade OpenFlow virtualization schemes (Skoldstrom and John, 2013; Koponen et al., 2014) such as virtualized broadband access infrastructures (Anwer et al., 2013), methods which offer network-as-a-service (Media Release, 2016).
- Programmable GEPON and DWDM ROADM (Parniewicz et al., 2014; Belter et al., 2014a, 2014b; Clegg et al., 2014).
- Large-scale inter-autonomous systems (ASs) SDN enabled deployments (Bennesby et al., 2012).
- Flexible control of network resources (NTT DATA, 2016) like offering MPLS services by means of an SDN approach (Das et al., 2011).
- Investigation of new network architectures including the ones for separating network edge from the core (Casado et al., 2012; Martinello et al., 2014), with the latter forming the fabric that transports packets as defined by an intelligent edge, to software-defined Internet exchange points (SDX, 2016; Stringer et al., 2013).

It is obvious that that SDN can be considered as an opportunity for telecom and cloud providers which results in flexibility, cost-efficacy, and better handling of their networks. In fact, some earlier ideas and theories regarding SDN have been realized but still many other open issues which were mentioned is this overview should be addressed in future studies.

6.8. SDN: A missing piece of software-defined puzzle

Converging different technologies facilitates the development of fully programmable IT infrastructures. At the present, the entire IT stack can be dynamically and automatically configured or reconfigured from the network infrastructure up to the applications so that workload changes can be better handled. The latest advances and developments make on-demand provisioning of resources at nearly all infrastructural layers possible. Recently, the automated provisioning and orchestration of IT infrastructures was labeled software-defined environments (SDEs) (Racherla et al., 2014; Li et al., 2014) by IBM. Indeed, it is a new method which is of remarkable significance in simplifying IT management, optimizing infrastructure use, reducing costs and time of new ideas and products. Workloads in SDE can be easily and automatically allocated to proper IT resources based on the followings: application characteristics, security and service level policies, the best available resources for continuous and dynamic optimization and reconfiguration of infrastructure issues in a rapid and responsive manner. It should be noticed that one missing key piece of SDE is software-defined networking. The followings are regarded as the four major building blocks of SDE (Arnold et al., 2014):

- Software-defined networks (SDN) (Dixon et al., 2014; Racherla et al., 2014).
- Software-defined storage (SDS) (Alba et al., 2014).
- Software-defined compute (SDC) (Li et al., 2014).
- Software-defined management (IBM Software Defined Environments, 2016) (SDM).

IBM SmartCloud orchestrator is known as the initial instances of SDE (Li et al., 2014; Racherla et al., 2014).

7. Conclusion

As discussed in the paper, handling traditional networks is a complex and challenging task which is partially attributed to the fact that control and data planes are vertically integrated and vendor specific. The popularity of SDN is increasingly enhanced thanks to the interesting features it offers by providing innovations with regard to design, organization and management of the networks. Some of the outstanding concepts of SDN are: dynamic programmability in forwarding devices through open southbound interfaces, decoupling control and data plane and the global view of the network which is due to logical centralization. However, it should be noted that there are still many open research questions and gaps which need to be solved so that successful SDN can be accomplished.

In this paper, a comprehensive overview of programmable networks, i.e. the emerging field of Software-Defined Networking (SDN) was given. SDN architecture and its three planes, namely data plane, control plane and application plane were discussed in detail. Moreover, OpenFlow which was the standard protocol for control and data planes were described. It should be noted that the majority of related works have focused on the structure of SDN, control plane and OpenFlow. However, we not only investigated them but also we provided an exhaustive categorization of the state-of-the-art SDN technologies in all the three planes. Also, current SDN implementations, testing platforms and current standardization efforts were mentioned in this paper. Furthermore, we examined network services and applications based on a range of SDN paradigms from heterogeneous networks to ICN. As mentioned earlier in the paper, as promising research domain, SDN should be optimized in all three planes so that it can be successfully applied in industry and other required settings. Addressing the above-mentioned directions for further research can help sort out the existing challenges and improve SDN.

References

Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W., 2014. A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Netw. 71, 1–30.

Abolhasan, M., Lipman, J., Ni, W., Hagelstein, B., 2015. Software-defined wireless networking: centralized, distributed, or hybrid? Netw., IEEE 29 (4), 32–38.

Akyildiz, I.F., Wang, P., Lin, S.-C., 2015. SoftAir: A software defined networking architecture for 5G wireless systems. Comput. Netw. 85, 1–18.

Ali, S.T., Sivaraman, V., Radford, A., Jha, S., 2015. A survey of securing networks using software defined networking. Reliab., IEEE Trans. 64 (3), 1086–1097.

ANDO, S., NAKAO, A., 2014. L7Classifier: packet classification applying regular expression to packet payload. Tech. Comm. Commun. Qual. 113 (405), 35–40.

Ahmed, M., Huici, F., Jahanpanah, A., 2012. Enabling dynamic network processing with clickos. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 293–294.

Agarwal, S., Kodialam, M., Lakshman T., 2013. Traffic engineering in software defined networks 2013 In: Proceedings of the IEEE INFOCOM, pp. 2211–2219.

Apache Thrift - Home: tree docs; 2016. Available from: ⟨http://thrift.apache.org/⟩.

Akhunzada, A., Gani, A., Anuar, N.B., Abdelaziz, A., Khan, M.K., Hayat, A., et al., 2015. Secure and dependable software defined networks. J. Netw. Comput. Appl.

Antonenko, V., Smelyanskiy, R., 2013. Global network modelling based on mininet approach. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 145–146.

Al-Shaer, E., Al-Haj S., 2010. FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures. In: Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration, ACM, pp. 37–44.

Active IRTF research groups, 2016. Available from: ⟨https://datatracker.ietf.org/rg/⟩.

Apache Cloudstack, 2016. Available from: ⟨http://cloudstack.apache.org/⟩.

Azodolmolky, S., Nejabati, R., Escalona, E., Jayakumar, R., Efstathiou, N., Simeonidou, D., 2011. Integrated OpenFlow–GMPLS control plane: an overlay model for software defined packet over optical networks. Opt Express 19 (26), B421–B428.

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B., 2012. A survey of information-centric networking. Commun. Mag., IEEE 50 (7), 26–36.

Appelman, M., de Boer, M., 2012. Performance Analysis of OpenFlow Hardware Tech Rep.. University of Amsterdam.

Agarwal, K., Dixon, C., Rozner, E., Carter, J., 2014. Shadow MACs: scalable label-switching for commodity ethernet. In: Proceedings of the Third Workshop on

Hot Topics in Software Defined Networking. ACM, pp. 157–162.
Anwer, B., Benson, T., Feamster, N., Levin, D., Rexford, J., 2013. A slick control plane for network middleboxes. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 147–148.
Arnold, W.C., Arroyo, D., Segmuller, W., Spreitzer, M., Steinder, M., Tantawi, A.N., 2014. Workload orchestration and optimization for software defined environments. IBM J. Res. Dev. 58 (2/3) 11: 1-: 2.
Alba, A., Alatorre, G., Bolik, C., Corrao, A., Clark, T., Gopisetty, S., et al., 2014. Efficient and agile storage management in software defined environments. IBM J. Res. Dev. 58 (2/3) 5: 1-5: 12.
Brocade MLX Series - Brocade 2016. Available from: ⟨http://www.brocade.com/en/products-services/routers/mlx-series.html⟩.
Bosshart, P., Gibb, G., Kim, H.-S., Varghese, G., McKeown, N., Izzard, M., et al., 2013. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. ACM SIGCOMM Comput. Commun. Rev. 43 (4), 99–110.
Bianco, A., Birke, R., Giraudo, L., Palacin, M., 2010. Openflow switching: data plane performance. Commun. (ICC) 2010, 1–5.
Bianchi, G., Bonola, M., Capone, A., Cascone, C., 2014. OpenState: programming platform-independent stateful openflow applications inside the switch. ACM SIGCOMM Comput. Commun. Rev. 44 (2), 44–51.
Benton, K., Camp, L.J., Small, C., 2013. Openflow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 151–152.
Belter, B., Binczewski, A., Dombek, K., Juszczyk, A., Ogrodowczyk, L., Parniewicz, D., et al., 2014. Programmable abstraction of datapath. In: Proceedings of the 2014 Third European Workshop on Software Defined Networks (EWSDN), pp. 7–12.
Bozakov, Z., Papadimitriou, P., 2012. Autoslice: automated and scalable slicing for software-defined networks. In: Proceedings of the 2012 ACM Cconference on CoNEXT Student Workshop, pp. 3–4.
Bennesby, R., Fonseca, P., Mota, E., Passito, A., 2012. An inter-as routing component for software-defined networks. In: Proceedings of the Network Operations and Management Symposium (NOMS), IEEE, pp. 138–145.
Braga, R., Mota, E., Passito, A., 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proceedings of the 35th Conference on Local Computer Networks (LCN), IEEE, pp. 408–415.
Bansal, M., Mehlman, J., Katti, S., Levis, P., 2012. Openradio: a programmable wireless dataplane. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 109–114.
Baskett, P., Shang, Y., Zeng, W., Guttersohn, B., 2013. SDNAN: Software-defined networking in ad hoc networks of smartphones. In: Proceedings of the Consumer Communications and Networking Conference (CCNC), IEEE, pp. 861–862.
Brief OS, 2013. OpenFlow™-Enabled Mobile and Wireless Networks. September.
Benson, T., Akella, A., Shaikh, A., Sahu, S., 2011. CloudNaaS: a cloud networking platform for enterprise applications. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, pp. 8.
Braun, W., Menth, M., 2014. Wildcard compression of inter-domain routing tables for OpenFlow-based software-defined networking. In: Proceedings of the Third European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 25–30.
Bosshart, P., Gibb, G., Kim, H.-S., Varghese, G., McKeown, N., Izzard, M., et al., 2013. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN, ACM SIGCOMM Computer Communication Review. ACM, pp. 99–110.
Botelho, F., Ramos, V., Manuel, F., Kreutz, D., Bessani, A., 2013. On the feasibility of a consistent and fault-tolerant data store for SDNs. In: Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 38–43.
Botelho, F.,Bessani, A., Ramos , F., Ferreira, P., 2014. On the design of practical fault-tolerant SDN controllers. In: Proceedings of the Third European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 73–78.
Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., et al., 2014. ONOS: towards an open, distributed SDN OS. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ACM, pp. 1–6.
Belter, B., Parniewicz, D., Ogrodowczyk, L., Binczewski, A., Stroinski, M., Fuentes, V., et al., 2014. Hardware abstraction layer as an SDN-enabler for non-OpenFlow network equipment. In: Proceedings of the Third European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 117–118.
Cao, B., He, F., Li, Y., Wang, C., Lang, W., 2015. Software defined virtual wireless network: framework and challenges. Netw., IEEE 29 (4), 6–12.
CPqD/ofsoftswitch13: @github; 2016. Available from: ⟨https://github.com/CPqD/ofsoftswitch13⟩.
Centec Networks - SDN/OpenFlow Switch - V330: CodeDevelopment Platform, 2016. Available from: ⟨http://www.centecnetworks.com/en/SolutionList.asp?ID=42⟩.
Consortium OS, 2016. OpenFlow Switch Consortium, OpenFlow Switch Specification Version 1.0. 0. Available from: ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf⟩.
Crockford, D., 2006. The application/json Media Type for JavaScript Object Notation (JSON). [Updated July, 2006]. Available from: ⟨https://tools.ietf.org/html/rfc4627⟩.
Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S., 2011. DevoFlow: Scaling flow management for high-performance networks. ACM SIGCOMM Comput. Commun. Rev. 41 (4), 254–265.
Canini, M., Venzano, D., Peresini, P., Kostic, D., Rexford, J., 2012. A NICE way to test OpenFlow applications. NSDI 12, 127–140.
Ciuffo, L., 2016. FIBRE. Available from: ⟨http://www.fibre-ict.eu/⟩.
Civanlar, S., Parlakisik, M., Gorkemli, B., Kaytaz, B., Onem, E., 2010. A qos-enabled openflow environment for scalable video streaming. In: GLOBECOM Workshops (GC Wkshps), IEEE, pp. 351–356.
Choi, Y. Implementation of Content-oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure.
Calvert, K.L., Edwards, W.K., Feamster, N., Grinter, R.E., Deng, Y., Zhou, X., 2011. Instrumenting home networks. ACM SIGCOMM Comput. Commun. Rev. 41 (1), 84–89.
Clegg, R.G., Spencer, J., Landa, R., Thakur, M., Mitchell, J., Rio, M., 2014. Pushing software defined networking to the access. In: Proceedings of the Third European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 31–36.
Casado, M., Koponen, T., Shenker, S., Tootoonchian, A., 2012. Fabric: a retrospective on evolving SDN. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 85–90.
Dobrescu, M., Egi, N., Argyraki, K., Chun, B.-G., Fall, K., Iannaccone, G., et al., 2009. RouteBricks: exploiting parallelism to scale software routers. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 15–28.
Drutskoy, D., Keller, E., Rexford, J., 2013. Scalable network virtualization in software-defined networks. Internet Interoper. Comput., IEEE 17 (2), 20–27.
Dely, P., Kassler, A., Bayer, N., 2011. Openflow for wireless mesh networks. In: Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), IEEE, pp. 1–6.
Desai, M., Nandagopal, T., 2010. Coping with link failures in centralized control plane architectures. In: Proceedings of the Second International Conference on Communication Systems and Networks (COMSNETS), IEEE, pp. 1–10.
Das, S., Sharafat, A., Parulkar, G., McKeown, N., 2011. MPLS with a simple OPEN control plane. In: Proceedings of the Optical Fiber Communication Conference, Optical Society of America, OWP2.
Dixon, C., Olshefski, D., Jain, V., DeCusatis, C., Felter, W., Carter, J., et al., 2014. Software defined networking to support the software defined environment. IBM J. Res. Dev. 58 (2/3) 3: 1-3: 14.
Extreme Networks, "Blackdiamond x8," 2016. Available from: ⟨http://www.extremenetworks.com/libraries/products/DSBDX1832.pdf⟩.
EZchip | Network Processors 2016. Available from: ⟨http://www.ezchip.com/⟩.
EstiNet Technologies, 2016. Available from: ⟨http://www.estinet.com/products.php⟩.
ETSI – European Telecommunications Standards Institute, 2016. Available from: ⟨http://www.etsi.org/⟩.
Egilmez, H.E., Dane, S.T., Bagci, K.T., 2012. OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. In: Proceedings of the Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), Asia-Pacific, IEEE, pp. 1–8.
Farhady, H., Lee, H., Nakao, A., 2015. Software-defined networking: a survey. Comput. Netw. 81, 79–95.
FlowForwarding/LINC-Switch: @github, 2016. Available from: ⟨https://github.com/FlowForwarding/LINC-Switch⟩.
Farhadi, H., Du, P., Nakao, A., 2014. User-defined actions for SDN. In: Proceedings of The Ninth International Conference on Future Internet Technologies, pp. 3.
Foster, N., Harrison, R., Freedman, M.J., Monsanto, C., Rexford, J., Story, A., et al., 2011. Frenetic: a network programming language. ACM SIGPLAN Not. 46 (9), 279–291.
Foster, N., Freedman, M.J., Harrison, R., Rexford, J., Meola, M.L., Walker, D., 2010. Frenetic: a high-level language for OpenFlow networks. In: Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow, ACM, pp. 6.
Fayazbakhsh, S.K., Sekar, V., Yu, M., Mogul, J.C., 2013. Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 19–24.
Farhad, H., Lee, H., Nakao, A., 2014. Data Plane Programmability in SDN. In: Proceedings of the IEEE 22nd International Conference on Network Protocols (ICNP), pp. 583–588.
Ferguson, A.D., Guha, A., Liang, C., Fonseca, R., Krishnamurthi, S., 2013. Participatory networking: an API for application control of SDNs, ACM SIGCOMM Computer Communication Review. ACM, pp. 327–338.
Feamster, N., Rexford, J., Zegura, E., 2014. The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Comput. Commun. Rev. 44 (2), 87–98.
floodlight/loxigen: @github, 2016. Available from: ⟨https://github.com/floodlight/loxigen⟩.
Gill, H., Lin, D., Sarna, L., Mead, R., Lee, K.C., Loo, B.T., 2012. SP4: scalable programmable packet processing platform. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 75–76.
Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., et al., 2008. NOX: towards an operating system for networks. ACM SIGCOMM Comput. Commun. Rev. 38 (3), 105–110.
GlobalNOC, 2016. Available from: ⟨http://globalnoc.iu.edu/sdn/oess.html⟩.
GENI, 2016. Available from: ⟨http://www.geni.net/⟩.
Gember, A., Prabhu, P., Ghadiyali, Z., Akella, A., 2012. Toward software-defined middlebox networking. In: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ACM, pp. 7–12.
Ghorbani, S., Caesar, M., 2012. Walk the line: consistent network updates with bandwidth guarantees. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 67–72.
Gudipati, A., Perry, D., Li, L.E., Katti, S., 2013. SoftRAN: Software defined radio access network. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot

Topics in Software Defined Networking, ACM, pp. 25–30.
Ghaffari, A., 2015. Congestion control mechanisms in Wireless Sensor networks: a survey. J. Netw. Comput. Appl. 52, 101–115.
Guimaraes, C., Corujo, D., Aguiar, R.L., Silva, F., Frosi, P., 2013. Empowering software defined wireless Networks through Media Independent Handover management. In: Proceedings of the Global Communications Conference (GLOBECOM), IEEE, pp. 2204–2209.
Gringeri, S., Bitar, N., Xia, T.J., 2013. Extending software defined network principles to include optical transport. Commun. Mag., IEEE 51 (3), 32–40.
Gudla, V.R., Das, S., Shastri, A., Parulkar, G., McKeown, N., Kazovsky, L., et al., 2010. Experimental demonstration of OpenFlow control of packet and circuit switches. In: Proceedings of the Optical Fiber Communication Conference, Optical Society of America, OTuG2.
Hu, F., Hao, Q., Bao, K., 2014. A survey on software-defined network and openflow: from concept to implementation. Commun. Surv. Tutor., IEEE 16 (4), 2181–2206.
Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C., Gayraud, T., 2014. Software-defined networking: challenges and research opportunities for future internet. Comput. Netw. 75, 453–471.
Huang, H., Li, P., Guo, S., Zhuang, W., 2015. Software-defined wireless mesh networks: architecture and traffic orchestration. Netw., IEEE 29 (4), 24–30.
HP, HP 8200 ZL switch series, 2016. Available from: ⟨http://h17007.www1.hp.com/us/en/networking/products/switches/HP8200zlSwitchSeries/⟩.
Huawei Technologies Co., Ltd., 2016. Cx600 metro services platform. Available from: ⟨http://www.huawei.com/ucmf/groups/public/documents/attachments/hw_132369.pdf⟩.
Han, S., Jang, K., Park, K., Moon, S., 2011. PacketShader: a GPU-accelerated software router. ACM SIGCOMM Comput. Commun. Rev. 41 (4), 195–206.
Hassas Yeganeh, S., Ganjali, Y., 2012. Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 19–24.
Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 7–12.
Hu, Y., Wang, W., Gong, X., Que, X., Cheng, S., 2014. On reliability-optimized controller placement for Software-Defined Networks. Communications, China 11 (2), 38–54.
Home – Beacon – Confluence 2016. Available from: ⟨https://openflow.stanford.edu/display/Beacon/Home⟩.
Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., McKeown, N., 2012. Reproducible network experiments using container-based emulation. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 253–264.
Handigol, N., Heller, B., Jeyakumar, V., Maziéres, D., McKeown, N., 2012. Where is the debugger for my software-defined network? In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 55–60.
Hinrichs, T.L., Gude, N.S., Casado, M., Mitchell, J.C., Shenker, S., 2009. Practical declarative network management. In: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, ACM, pp. 1–10.
Hares, S., Nadeau, T., Halpern, J., Ward, D., Atlas, A., 2013. An Architecture for the Interface to the Routing System. Available from: ⟨https://tools.ietf.org/html/draft-ietf-i2rs-architecture-05⟩.
Home – IEEE Software Defined Networks, 2016. Available from: ⟨http://sdn.ieee.org/⟩.
Home » OpenStack Open Source Cloud Computing Software: Code, 2016. Available from: ⟨http://www.openstack.org/⟩.
Handigol, N., Flajslik, M., Seetharaman, S., McKeown, N., Johari, R., 2010. Aster* x: Load-balancing as a network primitive. In: Proceedings of the 9th GENI Engineering Conference (Plenary), pp. 1–2.
Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., Johari, R., 2009. Plug-n-Serve: load-balancing web traffic using OpenFlow. ACM SIGCOMM Demo 4 (5), 6.
Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., et al., 2010. ElasticTree: saving energy in data center networks. NSDI 10, 249–264.
Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., et al., 2013. Achieving high utilization with software-driven WAN, ACM SIGCOMM Computer Communication Review. ACM, pp. 15–26.
IRIS: The Recursive SDN-Openflow Controller by ETRI 2016. Available from: ⟨http://openiris.etri.re.kr/⟩.
Internet2 2016. Available from: ⟨http://www.internet2.edu/⟩.
Internet Engineering Task Force (IETF), 2016. Available from: ⟨https://www.ietf.org/⟩.
IRTF Software-Defined Networking Research Group (SDNRG), 2016. Available from: ⟨https://irtf.org/sdnrg⟩.
ITU Telecommunication, 2016. Available from: ⟨http://www.itu.int/en/ITU-T/Pages/default.aspx/⟩.
IEEE, 2016. Available from: ⟨https://www.ieee.org/index.html⟩.
IBM Software Defined Environments, 2016. Available from: ⟨http://www-935.ibm.com/services/us/en/it-services/systems/server-services/software-defined-environment/⟩.
Jagadeesan, N.A., Krishnamachari, B., 2014. Software-defined networking paradigms in wireless networks: a survey. ACM Comput. Surv. (CSUR) 47 (2), 27.
Juniper/contrail-vrouter: @github, 2016. Available from: ⟨https://github.com/Juniper/contrail-vrouter⟩.
Juniper Networks, "Ex9200 ethernet switch," 2016. Available from: ⟨http://www.juniper.net/us/en/local/pdf/datasheets/1000432-en.pdf⟩.
Jeyakumar, V., Alizadeh, M., Kim, C., Mazières, D., 2013. Tiny packet programs for low-latency network control and monitoring. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, pp. 8.
SSLShader: Cheap SSL Acceleration with Commodity Processors. In: Jang, K., Han, S., Han, S., Moon, S.B., Park, K. (Eds.), NSDI.
JGN-X Website, 2016. Available from: ⟨http://www.jgn.nict.go.jp/english/index.html⟩.
Joint Coordination Activity on Software-Defined Networking (JCA-SDN), 2016. Available from: ⟨http://www.itu.int/en/ITU-T/jca/sdn/Pages/default.aspx⟩.
Jafarian, J.H., Al-Shaer, E., Duan, Q., 2012. Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 127–132.
Jin, X., Li, L.E., Vanbever, L., Rexford, J., 2013. Softcell: Scalable and flexible cellular core network architecture. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, ACM, pp. 163–174.
Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., et al., 2013. B4: Experience with a globally-deployed software defined WAN, ACM SIGCOMM Computer Communication Review. ACM, pp. 3–14.
Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P., 2011. Modeling and performance evaluation of an OpenFlow architecture. In: Proceedings of the 23rd International Teletraffic Congress: International Teletraffic Congress, pp. 1–7.
Jaxon, 2016. Available from: ⟨http://jaxon.onuos.org/⟩.
Kobayashi, M., Seetharaman, S., Parulkar, G., Appenzeller, G., Little, J., Van Reijendam, J., et al., 2014. Maturing of OpenFlow and Software-defined Networking through deployments. Comput. Netw. 61, 151–175.
Kogan, K., Nikolenko, S., Culhane, W., Eugster, P., Ruan, E., 2013. Towards efficient implementation of packet classifiers in SDN/OpenFlow. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 153–154.
Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F., 2000. The Click modular router. ACM Trans. Comput. Syst. (TOCS) 18 (3), 263–297.
Kim, N., Yoo, J.-Y., Kim, N.L., Kim, J., 2012. A programmable networking switch node with in-network processing support. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 6611–6615.
Kang, N., Reich, J., Rexford, J., Walker, D., 2012. Policy transformation in software defined networks. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 309–310.
Kanizo, Y., Hay, D., Keslassy, I., 2013. Palette: distributing tables in software-defined networks. In: Proceedings IEEE INFOCOM, pp. 545–549.
Kang, N., Liu, Z., Rexford, J., Walker, D., 2013. Optimizing the one big switch abstraction in software-defined networks. In: Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, pp. 13–24.
Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., et al., 2010. Onix: a distributed control platform for large-scale production networks. OSDI 10, 1–6.
Katta, N.P., Rexford, J., Walker, D., 2013. Incremental consistent updates. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 49–54.
Kissel, E., Fernandes, G., Jaffee, M., Swany, M., Zhang, M., 2012. Driving software defined networks with xsp. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 6616–6621.
Kuźniar, M., Perešíni, P., Vasić, N., Canini, M., Kostić, D., 2013. Automatic failure recovery for software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 159–160.
Kozat, U.C., Liang, G., Kokten, K., 2013. Verifying forwarding plane connectivity and locating link failures using static rules in software defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 157–158.
Kempf, J., Bellagamba, E., Kern, A., Jocha, D., Takács, A., Sköldström, P., 2012. Scalable fault management for OpenFlow. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 6606–6610.
Klein, D., Jarschel, M., 2013. An OpenFlow extension for the OMNeT++ INET framework. In: Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 322–329.
Khurshid, A., Zhou, W., Caesar, M., Godfrey, P., 2012. Veriflow: verifying network-wide invariants in real time. ACM SIGCOMM Comput. Commun. Rev. 42 (4), 467–472.
Kazemian, P., Varghese, G., McKeown, N., 2012. Header space analysis: static checking for networks. NSDI, 113–126.
Kim, W., Sharma, P., Lee, J., Banerjee, S., Tourrilhes, J., Lee, S.-J., et al., 2010. Automated and scalable QoS control for network convergence. Proc. INM/WREN 10, 1.
Kim, H., Feamster, N., 2013. Improving network management with software defined networking. Commun. Mag., IEEE 51 (2), 114–119.
Kim, H., Reich, J., Gupta, A., Shahbaz, M., Feamster, N., Clark, R., 2015. Kinetic: Verifiable dynamic network control. In: Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), pp. 59–72.
Keller, E., Ghorbani, S., Caesar, M., Rexford, J., 2012. Live migration of an entire network (and its hosts). In: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ACM, pp. 109–114.
Kannan, K., Banerjee, S., 2012. Scissors: Dealing with header redundancies in data centers through SDN. In: Proceedings of the 8th International Conference on Network and Service Management. International Federation for Information Processing, pp. 295–301.

Kreutz, D., Ramos, F., Verissimo, P., 2013. Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 55–60.
Kreutz, D., Casimiro, A., Pasin, M., 2012. A trustworthy and resilient event broker for monitoring cloud infrastructures. Distrib. Appl. Interoper. Syst., 87–95.
Kim, H., Santos, J.R., Turner, Y., Schlansker, M., Tourrilhes, J., Feamster, N., 2012. Coronet: Fault tolerance for software defined networks. In: Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP), IEEE, pp. 1–2.
Koponen, T., Amidon, K., Balland, P., Casado, M., Chanda, A., Fulton, B., et al. (Eds.), 2014. Network Virtualization in Multi-Tenant Datacenters. USENIX NSDI.
Luo, Y., Cascon, P., Murray, E., Ortega, J., 2009. Accelerating OpenFlow switching with network processors. In: Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 70–71.
Lee, J., Tourrilhes, J., Sharma, P., Banerjee, S., 2011. No more middlebox: integrate processing into network. ACM SIGCOMM. Comput. Commun. Rev. 41 (4), 459–460.
Lu, G., Miao, R., Xiong, Y., Guo, C., 2012. Using cpu as a traffic co-processing unit in commodity switches. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 31–36.
Liu, Y., Li, Y., Wang, Y., Yuan, J., 2015. Optimal scheduling for multi-flow update in Software-Defined Networks. J. Netw. Comput. Appl. 54, 11–19.
Li, L.E., Mao, Z.M., Rexford, J., 2012. Toward software-defined cellular networks. In: Proceedings of the European Workshop on Software Defined Networking (EWSDN), IEEE, pp. 7–12.
Luo, T., Tan, H.-P., Quek, T.Q., 2012. Sensor OpenFlow: Enabling software-defined wireless sensor networks. Commun. Lett., IEEE 16 (11), 1896–1899.
Liu, L., Tsuritani, T., Morita, I., Guo, Wu, J., 2011. OpenFlow-based wavelength path control in transparent optical networks: a proof-of-concept demonstration. In: Proceedings of the European Conference and Exposition on Optical Communications. Optical Society of America, pp. Tu. 5. K. 2.
Liu, Z., Li, Y., Su, L., Jin, D., Zeng, L., 2013. M2cloud: software defined multi-site data center network control framework for multi-tenant, ACM SIGCOMM Computer Communication Review. ACM, pp. 517–518.
Lin, P., Hart, J., Krishnaswamy, U., Murakami, T., Kobayashi, M., Al-Shabibi, A., et al., 2013. Seamless Interworking of SDN and IP. ACM SIGCOMM Computer Communication Review. ACM, pp. 475–476.
Levin, D., Canini, M., Schmid, S., Feldmann, A., 2013. Incremental SDN deployment in enterprise networks. ACM SIGCOMM Computer Communication Review. ACM, pp. 473–474.
Li, C., Brech, B., Crowder, S., Dias, D.M., Franke, H., Hogstrom, M., et al., 2014. Software defined environments: An introduction. IBM J. Res. Dev. 58 (2/3), 1–11.
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., et al., 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.
Macedo, D.F., Guedes, D., Vieira, L.F., Vieira, M.A., Nogueira, M., 2015. Programmable networks—from software-defined radio to software-defined networking. Commun. Surv. Tutor., IEEE 17 (2), 1102–1125.
Mizrahi, T., Moses, Y., 2013. Time-based updates in software defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 163–164.
Mogul, J.C., Congdon, P., 2012. Hey, you darned counters!: get off my ASIC! In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 25–30.
Monsanto, C., Reich, J., Foster, N., Rexford, J., Walker, D., 2013. Composing Software Defined Networks. NSDI, pp. 1–13.
Mccauley, J. Pox, 2016. A Python-Based Openflow Controller. Available from: ⟨http://www.noxrepo.org/pox/about-pox/⟩.
Mul, 2016. Available from: ⟨http://sourceforge.net/p/mul/wiki/Home/⟩.
MirageOS, 2015. Available from: ⟨https://mirage.io/⟩.
MEF, 2016. Available from: ⟨http://www.mef.net/⟩.
Macapuna, C.A., Rothenberg, C.E., Magalhães, Mc.F., 2010. In-packet Bloom filter based data center networking with distributed OpenFlow controllers. In: GLOBECOM Workshops (GC Wkshps), IEEE, pp. 584–588.
Mehdi, S.A., Khalid, J., Khayam, S.A., 2011. Revisiting traffic anomaly detection using software defined networking. Recent Adv. Intrusion Detect., 161–180.
Mortier, R., Rodden, T., Lodge, T., McAuley, D., Rotsos, C., Moore, A.W., et al., 2012. Control and understanding: owning your home network. In: Proceedings of the Fourth International Conference on Communication Systems and Networks (COMSNETS), IEEE, pp. 1–10.
Mann, V., Kannan, K., Vishnoi, A., Iyer, A.S., 2013. Ncp: Service replication in data centers through software defined networking. In: Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), IEEE, pp. 561–567.
Melazzi, N.B., Detti, A., Mazza, G., Morabito, G., Salsano, S., Veltri, L., 2012. An openflow-based testbed for information centric networking. Future Netw Mob Summit (FutureNetw), 1–9.
Media Release: Pacnet Offers First Pan-Asia Network-as-a-Service Architecture, 2016. Available from: ⟨http://www.cmo.com.au/mediareleases/17701/pacnet-offers-first-pan-asia-network-as-a-service/⟩.
Martinello, M., Ribeiro, M., de Oliveira, R.E.Z., de Angelis Vitoi, R., 2014. Keyflow: a prototype for evolving SDN toward core network fabrics. Netw., IEEE 28 (2), 12–19.
Nunes, B., Mendonca, M., Nguyen, X.-N., Obraczka, K., Turletti, T., 2014. A survey of software-defined networking: past, present, and future of programmable networks. Commun. Surv. Tutor., IEEE 16 (3), 1617–1634.
Networks A. Arista Networks, "7150 series," @AristaNetworks, 2016. Available from: ⟨http://www.aristanetworks.com/media/system/pdf/Datasheets/7150S_Datasheet.pdf⟩.
Networking | Systems: Lenovo Corporation, 2016. Available from: ⟨http://shop.lenovo.com/us/en/systems/networking/⟩.
Nec. NEC ProgrammableFlow Networking, 2016. Available from: ⟨https://www.necam.com/SDN/⟩.
Nakao, A. (Ed.), 2012. FLARE: Open deeply programmable switch. In: Proceedings of the 16th GENI Engineering Conference.
Nakao, A., 2012. VNode: a deeply programmable network testbed through network virtualization. In: proceedings of the 3rd IEICE Technical Committee on Network Virtualization.
Nishida, Y., Nakao, A., 2012. In-network ad-targeting through wifi ap virtualization. In: Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT), pp. 1092–1097.
Ng, E., 2010. Maestro: A System for Scalable OpenFlow Control TSEN Maestro-Technical Report TR10-08. Rice University.
NodeFlow: An OpenFlow Controller Node Style | garyberger.net, 2016. Available from: ⟨http://garyberger.net/?p=537⟩.
Nascimento, M.R., Rothenberg, C.E., Salvador, M.R., Corrêa, C.N., de Lucena, S.C., Magalhães, M.F., 2011. Virtual routers as a service: the routeflow approach leveraging software-defined networks. In: Proceedings of the 6th International Conference on Future Internet Technologies, pp. 34–37.
NEC Global, 2016. Available from: ⟨http://www.nec.com/⟩.
Nakagawa, Y., Hyoudou, K., Shimizu, T., 2012. A management method of IP multicast in overlay networks using openflow. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 91–96.
Nakao Akihiro, "WiVi: Wi-Fi Network Virtualization Infrastructure", 2016. Available from: ⟨http://www.ieice.org/~nv/NV-Sept_10-nakao.pdf⟩.
Nguyen, X.N., Saucez, D., Turletti, T., 2013. Efficient caching in content-centric networks using OpenFlow. In: IEEE INFOCOM 2013 Workshop, pp. 1–2.
Naudts, B., Kind, M., Westphal, F.-J., Verbrugge, S., Colle, D., Pickavet, M., 2012. Techno-economic analysis of software defined networking as architecture for the virtualization of a mobile network. In: Proceedings of the European Workshop on Software Defined Networking (EWSDN), IEEE, pp. 67–72.
NTT DATA's Advance in SDN Business Provides Highly-Flexible Control of Network by Software | NTT DATA Global, 2016.
ns-3 2016. Available from: ⟨https://www.nsnam.org/⟩.
Open vSwitch, 2016. Available from: ⟨http://openvswitch.org/⟩.
OpenFlow, 2016. Available from: ⟨http://archive.openflow.org/wp/downloads/⟩.
OpenFlowClick - OpenFlow Wiki, 2016. Available from: ⟨http://archive.openflow.org/wk/index.php/OpenFlowClick⟩.
OpenFlow Switch Consortium, 2016. "OpenFlow Switch Specification Version 1.2.0". Available from: ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.0.pdf⟩.
OpenFlow Switch Consortium, 2016. "OpenFlow Switch Specification Version 1.3.0". Available from: ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf⟩.
OpenFlow Switch Consortium, 2016. "OpenFlow Switch Specification Version 1.4.0." 2016. Available from: ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf⟩.
OpenFlow Switch Consortium, 2016. "OpenFlow Switch Specification Version 1.5.0". Available from: ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf⟩.
ONF, 2014. "OpenFlow management and configuration protocol (OF-Config 1.1.1)," March. Available from: ⟨https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow-config/of-config-1-1-1.pdf⟩.
Open Network Operating System, 2016. Available from: ⟨http://www.slideshare.net/umeshkrishnaswamy/open-network-operating-system⟩.
Othman, M., Okamura, K., 2013. Hybrid control model for flow-based networks. In: Proceedings of the IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW), pp. 765–770.
Oflops - Cbench, 2016. Available from: ⟨http://archive.openflow.org/wk/index.php/Oflops⟩.
OFTestTutorial/-OpenFlow Wiki, 2016. Available from: ⟨http://archive.openflow.org/wk/index.php/OFTestTutorial/⟩.
OpenFlow support in PlanetLab | PlanetLabEurope, 2016. Available from: ⟨http://www.planet-lab.eu/openflow/⟩.
Ofelia – About OFELIA, 2016. Available from: ⟨http://www.fp7-ofelia.eu/about-ofelia/⟩.
Orbit, 2016.
ONF Overview – Open Networking Foundation, 2016. Available from: ⟨https://www.opennetworking.org/about/onf-overview⟩.
Owens, H., Durresi, A., 2013. Video over software-defined networking (vsdn). In: Proceedings of the 16th International Conference on Network-Based Information Systems (NBiS), IEEE, pp. 44–51.
Open Networking Foundation – Open Networking Foundation, 2016. Available from: ⟨https://www.opennetworking.org/index.php?option=com_content&view=category&layout=blog&id=39&Itemid=152⟨=en/⟩.
Ooka, A., Ata, S., Koide, T., Shimonishi, H., Murata, M., 2013. OpenFlow-based content-centric networking architecture and router implementation. Future Netw. Mob. Summit (FutureNetworkSummit), 1–10.
Pantou: OpenFlow 1.0 for OpenWRT – OpenFlow Wiki, 2016. Available from:

⟨http://archive.openflow.org/wk/index.php/OpenFlow_1.0_for_OpenWRT⟩.
Pica8 Xorplus, 2016. Available from: ⟨http://sourceforge.net/projects/xorplus/⟩.
Pica8, "Pica8 3920," 2016. Available from: ⟨http://www.pica8.org/documents/pica8-datasheet-64x10gbe-p3780-p3920.pdf⟩.
PlexxiInc. Plexxi, "Plexxi Switch 1," @PlexxiInc, 2016. Available from: ⟨http://www.plexxi.com/wp-content/themes/plexxi/assets/pdf/Plexxi Switch 1 Datasheet Dec 2012.pdf⟩.
Pfaff, B., Davie, B., 2013. The Open vSwitch Database Management Protocol. [updated December, 2013]. Available from: ⟨https://tools.ietf.org/html/rfc7047⟩.
Parniewicz, D., Doriguzzi Corin, R., Ogrodowczyk, L., Rashidi Fard, M., Matias, J., Gerola, M., et al., 2014. Design and implementation of an openflow hardware abstraction layer. In: Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing, ACM, pp. 71–76.
Pan, H., Guan, H., Liu, J., Ding, W., Lin, C., Xie, G., 2013. The FlowAdapter: enable flexible multi-table processing on legacy hardware. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 85–90.
Perešíni, P., Kuzniar, M., Vasić, N., Canini, M., Kostiū, D., 2013. OF. CPP: Consistent packet processing for OpenFlow. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 97–102.
Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G., 2012. A security enforcement kernel for OpenFlow networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, pp. 121–126.
Phemius, K., Bouet, M., 2013. Openflow: Why latency does matter. In: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 680–683.
Patel, A., Ji, P., Wang, T., 2013. Qos-aware optical burst switching in openflow based software-defined optical networks. In: Proceedings of the 17th International Conference on Optical Network Design and Modeling (ONDM), IEEE, pp. 275–280.
Park, S.H., Lee, B., You, J., Shin, J., Kim, T., Yang, S., RAON: Recursive abstraction of OpenFlow networks. In: Proceedings of the 2014 Third European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 115–116.
Pries, R., Jarschel, M., Goll, S., 2012. On the usability of OpenFlow in data center environments. In: Proceedings of the IEEE International Conference on Communications (ICC), IEEE, pp. 5533–5537.
Pentikousis, K., Wang, Y., Hu, W., 2013. Mobileflow: toward software-defined mobile networks. Commun. Mag., IEEE 51 (7), 44–53.
Qi, Y., Fong, J., Jiang, W., Xu, B., Li, J., Prasanna, V., 2010. Multi-dimensional packet classification on FPGA: 100 Gbps and beyond. Field-Programmable Technol (FPT) 2010, 241–248.
Qazi, Z.A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., Yu, M., 2013. SIMPLE-fying middlebox policy enforcement using SDN, ACM SIGCOMM Computer Communication Review. ACM, pp. 27–38.
Qazi, Z.A., Lee, J., Jin, T., Bellala, G., Arndt, M., Noubir, G., 2013. Application-awareness in SDN, ACM SIGCOMM Computer Communication Review. ACM, pp. 487–488.
Ryu, 2016. An Operating System for Software Defined Network. Available from: ⟨http://osrg.github.io/ryu/⟩.
Rotsos, C., Sarrar, N., Uhlig, S., Sherwood, R., Moore, A.W., 2012. OFLOPS: An open framework for OpenFlow switch evaluation. Passiv. Active Meas., 85–95.
Reitblatt, M., Canini, M., Guha, A., Foster, N., 2013. Fattire: Declarative fault tolerance for software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 109–114.
Ren, J., Pentikousis, K., Westphal, C., Liu, W., Wang, J. The Role of Virtualization in Information-centric Network Deployment. E-LETTER.
Risso, F., Cerrato, I., 2012. Customizing data-plane processing in edge routers. In: Proceedings of the European Workshop on Software Defined Networking (EWSDN), pp. 114–120.
Racherla, S., Cain, D., Irwin, S., Ljungstrøm, P., Patil, P., Tarenzio, A.M., 2014. Implementing IBM Software Defined Network for Virtual Environments. IBM Redbooks.
rlenglet/openfaucet: @github, 2016. Available from: ⟨https://github.com/rlenglet/openfaucet⟩.
Scott-Hayward, S., O'Callaghan, G., Sezer, S., 2013. Sdn security: a survey. Future Netw. Serv. (SDN4FNS) 2013, 1–7.
Switch Light, 2016. Available from: ⟨http://www.bigswitch.com/products/switch-light⟩.
Sivaraman, A., Winstein, K., Subramanian, S., Balakrishnan, H., 2013. No silver bullet: extending SDN to the data plane. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, pp. 19.
Shimamura, M., Ikenaga, T., Tsuru, M., 2013. A design and prototyping of in-network processing platform to enable adaptive network services. IEICE Trans. Inf. Syst. 96 (2), 238–248.
FRESCO: Modular Composable Security Services for Software-Defined Networks. In: Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G., Tyson, M. (Eds.), NDSS.
Suzuki, K., Sonoda, K., Tomizawa, N., Yakuwa, Y., Uchida, T., Higuchi, Y., et al., 2014. A survey on OpenFlow technologies. IEICE Trans. Commun. 97 (2), 375–386.
Song, H., 2013. Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 127–132.
Smith, M., Dvorkin, M., Laribi, Y., Pandey, V., Garg, P., Weidenbacher, N., 2014. OpFlex Control Protocol. Available from: ⟨https://datatracker.ietf.org/doc/draft-smith-opflex/⟩.
Sune, M., Alvarez, V., Jungel, T., Toseef, U., Pentikousis, K., 2014. An OpenFlow implementation for network processors. In: Proceedings of the Third European Workshop on Software Defined Networks (EWSDN), pp. 123–124.
Shin, S., Gu, G., 2013. Attacking software-defined networks: a first feasibility study. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 165–166.
SNAC, 2016. Available from: ⟨https://github.com/bigswitch/snac-nox/blob/master/src/include/openflow/openflow/nicira-ext.h⟩.
Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., et al., 2010. Carving research slices out of your production networks with OpenFlow. ACM SIGCOMM Comput. Commun. Rev. 40 (1), 129–130.
STS by ucb-sts, 2016. Available from: ⟨http://ucb-sts.github.io/sts/⟩.
Scott, R.C., Wundsam, A., Zarifis, K., Shenker, S., 2012. What, where, and when: Software fault localization for sdn Tech Rep UCB/EECS-2012-2178. EECS Department, University of California, Berkeley.
SURF | SURFnet SDN testbed, 2016. Available from: ⟨https://www.surf.nl/en/innovationprojects/the-open-programmable-network/software-defined-networking/surfnet-sdn-testbed/index.html⟩.
Schwarz, M.F., Rojas, M., Miers, C.C., Redigolo, F.F., Carvalho, T.C., 2013. Emulated and software defined networking convergence. In: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 700–703.
Sharma, P., Banerjee, S., Tandel, S., Aguiar, R., Amorim, R., Pinheiro, D., 2013. Enhancing network management frameworks with SDN-like control. In: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), IEEE, pp. 688–691.
Shirali-Shahreza, S., Ganjali, Y., 2013. Flexam: Flexible sampling extension for monitoring and security applications in openflow. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 167–168.
Sekar, V., Egi, N., Ratnasamy, S., Reiter, M.K., Shi, G., 2012. Design and implementation of a consolidated middlebox architecture. Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 323–336.
Suenaga, M., Otani, M., Tanaka, H., Watanabe, K., 2012. Opengate on OpenFlow: system outline. In: Proceedings of the 2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, 2012.
Suresh, L., Schulz-Zander, J., Merz, R., Feldmann, A., Vazao, T., 2012. Towards programmable enterprise WLANS with Odin. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 115–120.
Shirazipour, M., John, W., Kempf, J., Green, H., Tatipamula, M., 2012. Realizing packet-optical integration with SDN and OpenFlow 1.1 extensions. In: Proceedings of the IEEE International Conference on Communications (ICC), IEEE, pp. 6633–6637.
Sadasivarao, A., Syed, S., Pan, P., Liou, C., Lake, A., Guok, C., et al. Open transport switch: a software defined networking architecture for transport networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 115–120.
Salsano, S., Blefari-Melazzi, N., Detti, A., Morabito, G., Veltri, L., 2013. Information centric networking over SDN and OpenFlow: architectural aspects and experiments on the OFELIA testbed. Comput. Netw. 57 (16), 3207–3221.
Suh, J., Jung, H., Kwon, T., Choi, Y., 2012. C-flow: content-oriented networking over openflow. Open Netw. Summit.
Stephens, B., 2012. Designing Scalable Networks For Future Large Datacenters. Rice University.
Skoldstrom, P., John, W., 2013. Implementation and evaluation of a carrier-grade OpenFlow virtualization scheme. In: Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), IEEE, pp. 75–80.
SDX: A software-defined internet exchange, 2016.
Stringer, J.P., Fu, Q., Lorier, C., Nelson, R., Rothenberg, C.E., 2013. Cardigan: deploying a distributed routing fabric. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, pp. 169–170.
Tanyingyong, V., Hidell, M., Sjödin, P., 2010. Improving pc-based openflow switching performance. In: Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 13.
Tootoonchian, A., Ganjali, Y., 2010. HyperFlow: a distributed control plane for OpenFlow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, pp. 3.
Team M. Mininet: An Instant Virtual Network on your Laptop (or other PC)-Mininet 2016. Available from: ⟨http://mininet.org/⟩.
Teixeira, J., Antichi, G., Adami, D., Del Chiaro, A., Giordano, S., Santos, A., 2013. Datacenter in a box: test your SDN cloud-datacenter controller at home. In: Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), pp. 99–104.
Trema, 2016. Available from: ⟨https://trema.github.io/trema/⟩.
The California OpenFlow Testbed Network, 2016. Available from: ⟨http://cenic.org/network/cotn⟩.
Tsou, T., Shi, X., Huang, J., Wang, Z., Yin, X., 2012. Analysis of Comparisons between OpenFlow and ForCES (updated 12.03.12). Available from: ⟨https://tools.ietf.org/html/draft-wang-forces-compare-openflow-forces-00⟩.
The OpenDaylight Platform | OpenDaylight, 2016. Available from: ⟨https://www.opendaylight.org/⟩.
Takagiwa, K., Ishida, S., Nishi, H., 2013. SoR-based programmable network for future software-defined network. In: Proceedings of the 37th Annual Computer Software and Applications Conference (COMPSAC), IEEE, pp. 165–166.
Trivisonno, R., Guerzoni, R., Vaishnavi, I., Soldani, D., 2015. SDN-based 5G mobile networks: architecture, functions, procedures and backward compatibility.

Trans. Emerg. Telecommun. Technol. 26 (1), 82–92.
Torres, J., Ferraz, L., Duarte, O., 2012. Controller-based routing scheme for Named Data Network COPPE/UFRJ, Tech Rep.. Electric. Eng. Program.
Vanbever, L., Reich, J., Benson, T., Foster, N., Rexford, J., 2013. Hotswap: correct and efficient controller upgrades for software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 133–138.
Varga, A., Hornig, R., 2008. An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering); 2008. pp. 60.
Voellmy, A., Wang, J., Yang, Y.R., Ford, B., Hudak, P., 2013. Maple: Simplifying SDN programming using algorithmic policies, ACM SIGCOMM Computer Communication Review. ACM, pp. 87–98.
Voellmy, A., Kim, H., Feamster, N., 2012. Procera: a language for high-level reactive network control. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp. 43–48.
Veltri, L., Morabito, G., Salsano, S., Blefari-Melazzi, N., Detti, A., 2012. Supporting information-centric functionality in software defined networks. In: IEEE International Conference on Communications (ICC): IEEE, pp. 6645–6650.
Wang, W., Haas, R., Salim, J.H., Doria, A., Khosravi, H.M., 2010. Forwarding and Control Element Separation (ForCES) Protocol Specification (updated March, 2010). Available from: ⟨https://tools.ietf.org/html/rfc5810⟩.
Wang, R., Butnariu, D., Rexford, J., 2011. OpenFlow-Based Server Load Balancing Gone Wild, pp. 11–12.
Williams, D., Jamjoom, H., 2013. Cementing high availability in OpenFlow with RuleBricks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 139–144.
Wen, X., Chen, Y., Hu, C., Shi, C., Wang, Y., 2013. Towards a secure controller platform for openflow applications. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 171–172.
Wundsam, A., Levin, D., Seetharaman, S., Feldmann, A., (Eds.), 2011. OFRewind: Enabling Record and Replay Troubleshooting for Networks. In: Proceedings of the USENIX Annual Technical Conference, 2011.
Wang, Y., Chen, H., Wu, X., Shu, L., 2016. An energy-efficient SDN based sleep scheduling algorithm for WSNs. J. Netw. Comput. Appl. 59, 39–45.
Wang, G., Ng, T., Shaikh, A., 2012. Programming your network at run-time for big data applications Proceedings of the first workshop on Hot topics in software defined networks. ACM, pp. 103–108.
Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H., 2014. A survey on software-defined networking. Commun Surv Tutorials, IEEE 17 (1), 27–51.
Yu, M., Rexford, J., Freedman, M.J., Wang, J., 2011. Scalable flow-based networking with DIFANE. ACM SIGCOMM Comput Commun Rev 41 (4), 351–362.
Yeganeh, S.H., Tootoonchian, A., Ganjali, Y., 2013. On scalability of software-defined networking. Commun Magazine, IEEE 51 (2), 136–141.
Yu, M., Jose, L., Miao, R., 2013. Software Defined Traffic Measurement with OpenSketch. NSDI 13, 29–42.
Yap, K.-K., Kobayashi, M., Sherwood, R., Huang, T.-Y., Chan, M., Handigol, N., et al., 2010. OpenRoads: Empowering research in mobile networks. ACM SIGCOMM Comput Commun Rev, 125–126.
Yap, K.-K., Sherwood, R., Kobayashi, M., Huang, T.-Y., Chan, M., Handigol, N., et al., 2010. Blueprint for introducing innovation into wireless mobile networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures, ACM, pp. 25–32.
Yang, M., Li, Y., Jin, D., Su, L., Ma, S., Zeng, L., 2013. OpenRAN: a software-defined ran architecture via virtualization, ACM SIGCOMM Computer Communication Review. ACM, pp. 549–550.
Yang, M., Li, Y., Jin, D., Zeng, L., Wu, X., Vasilakos, A.V., 2014. Software-defined and virtualized future mobile and wireless networks: a survey. Mob. Netw. Appl. 20 (1), 4–18.
Yu, M., Wundsam, A., Raju, M., 2014. NOSIX: a lightweight portability layer for the SDN OS. ACM SIGCOMM Comput. Commun. Rev. 44 (2), 28–35.
Yin, H., Xie, H., Tsou, T., Lopez, D., Aranda, P., Sidi, R., 2012. SDNi: A message exchange protocol for software defined networks (SDNs) across multiple domains. In: Proceedings of the IETF draft, work in progress.
Z-Series Packet-Optical Transport Platforms | Cyan, 2015. Available from: ⟨http://staging.cyaninc.com/products/z-series-packet-optical⟩.
Zeng, H., Kazemian, P., Varghese, G., McKeown, N., 2012. Automatic test packet generation. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, pp. 241–252.
Zhang, J., Zhang, J., Zhao, Y., Yang, H., Yu, X., Wang, L., et al., 2013. Experimental demonstration of OpenFlow-based control plane for elastic lightpath provisioning in Flexi-Grid optical networks. Opt. Express 21 (2), 1364–1373.