Security in Software Defined Networks: A Survey

Ijaz Ahmad, Suneth Namal, Mika Ylianttila, *Senior Member, IEEE*, and Andrei Gurtov, *Senior Member, IEEE*

***Abstract*—Software defined networking (SDN) decouples the network control and data planes. The network intelligence and state are logically centralized and the underlying network infrastructure is abstracted from applications. SDN enhances network security by means of global visibility of the network state where a conflict can be easily resolved from the logically centralized control plane. Hence, the SDN architecture empowers networks to actively monitor traffic and diagnose threats to facilitates network forensics, security policy alteration, and security service insertion. The separation of the control and data planes, however, opens security challenges, such as man-in-the middle attacks, denial of service (DoS) attacks, and saturation attacks. In this paper, we analyze security threats to application, control, and data planes of SDN. The security platforms that secure each of the planes are described followed by various security approaches for network-wide security in SDN. SDN security is analyzed according to security dimensions of the ITU-T recommendation, as well as, by the costs of security solutions. In a nutshell, this paper highlights the present and future security challenges in SDN and future directions for secure SDN.**

***Index Terms*—SDN, OpenFlow, network security, SDN security, application plane, control plane, data plane.**

I. Introduction

SOFTWARE defined networking (SDN) has become one of the most important network architectures for simplifying network management and enabling innovation in communication networks. SDN decouples the network control from the data forwarding plane. The control plane is logically centralized and the forwarding plane is rendered simple to act on decisions from the control plane [1]. In SDN, new control functions can be implemented by writing software-based logic in the control plane which deploys the decision logic in the forwarding plane through standard interfaces. A network operating system (NOS) in the control plane maps the entire network to different services and applications that are implemented on top of the control plane.

OpenFlow [2] is the most accepted and widely used implementation architecture of SDN. In OpenFlow, network policies and services are implemented as OpenFlow applications which interact with the control plane through the north-bound API (application programming interface) of the control plane. The control plane functionalities are implemented in an OpenFlow controller which interacts with the data plane through the OpenFlow protocol (south-bound API). OpenFlow-based SDN applications are developed that use the underlying network infrastructure and deploy various functions at run-time. Hence, control of the network traffic is transferred from the infrastructure to the administrator. As a result, network operators will gain high levels of network control, automation and optimization with the help of SDN applications.

SDN enhances network security with the centralized control of network behavior, global visibility of the network state and run-time manipulation of traffic forwarding rules. The centralized nature of networking in SDN enables enforcing network-wide security policies and mitigates the risks of policy collision. A network security application (e.g., security monitoring application) can request flow samples through the controller from the datapath. After security analysis, the security application can redirect the data path elements to either block the traffic, reroute to security middle boxes or restrict the traffic within a particular network jurisdiction. Moreover, updating security policies in SDN requires updating the security applications or adding security modules to the controller platform, rather than changing the hardware or updating its firmware.

On the contrary, conventional networks comprise large sets of vendor-specific manually configurable devices spread across networks. These devices are hardwired with specific algorithms used to route, control and monitor data flow based on function-specific logic in each device. Hence, it is difficult to seamlessly combine them into a single domain along with all the proprietary protocols, applications, and interfaces. The result is that legacy network architectures lack global visibility of the network state and have difficulties in deploying and maintaining coherent network-wide policies. This complexity and weaknesses in integration make it worse for maintaining stable and robust network security. For example, changing or updating security policies in these systems in the wake of changes in traffic behavior or intrusions is practically unmanageable and costly.

Network security in legacy network architectures is considered as an add-on which relies on manually configurable perimeter-based solutions. To implement a high-level network security policy, network operators must configure each device using vendor-specific low-level commands. However, manual configurations of network security technologies such as firewalls, intrusion detection/prevention systems (IDS/IPS) and IPSec technologies on extended sets of devices are prone to configuration errors, intra- and inter- domain policy conflicts that result in serious security breaches and threats [3]. A quantitative study on firewall configuration errors [4] shows that corporate firewalls enforce rule sets that violate well-known security guidelines and result in security breaches due to manual low-level configurations in each of the devices.

Manuscript received January 7, 2015; revised July 5, 2015; accepted August 15, 2015. Date of publication August 27, 2015; date of current version November 18, 2015.

I. Ahmad, S. Namal, and M. Ylianttila are with the Department of Communications Engineering, University of Oulu, 90014 Oulu, Finland (e-mail: iahmad@ee.oulu.fi; namal@ee.oulu.fi; mika.ylianttila@oulu.fi)

A. Gurtov is with Helsinki Institute for Information Technology (HIIT) and Department of Computer Science, Aalto University, 00076 Aalto, Finland (e-mail: gurtov@hiit.fi).

Digital Object Identifier 10.1109/COMST.2015.2474118

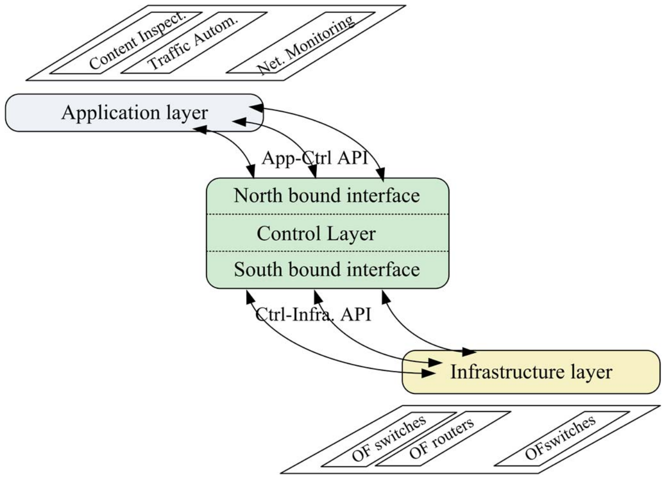

Fig. 1. The three planes/layers in the SDN architecture.

By logically centralizing the network control plane and introducing programmability, SDN enables security automation and run-time deployment of security procedures and policies. Network security systems leveraging from SDN can respond to network anomalies and spurious traffic conditions at runtime. To elaborate the functionality of the SDN architecture, the three main functional layers or SDN planes are presented in Fig. 1 and are constituted of:

- ***Application Plane***: It contains SDN applications for various functionalities, such as network management, policy implementation, and security services.
- ***Control Plane***: It is a logically centralized control framework that runs the NOS, maintains global view of the network, and provides hardware abstractions to SDN applications.
- ***Data Plane***: It is the combination of forwarding elements used to forward traffic flows based on instructions from the control plane.

Network security techniques can be implemented as applications in the SDN application plane. These applications would acquire the network state or resource information from the network control plane through the north-bound interface (App-Ctrl API). Similarly, security applications can collect samples of packets through the control plane. After security analysis, security applications/systems can redirect the traffic according to higher level security policies via the control plane using the south-bound API (Ctrl-Infra-API). Unlike traditional networks, data handling rules in SDN are implemented as software modules rather than embedding them in the hardware, thus, allowing run-time implementation of security policies and procedures.

However, SDN has its own challenges and limitations in terms of security, scalability, and supportability. Security has been on the forefront of these challenges. Since a centralized controller is responsible for managing the entire network, security compromise of the controller can render the whole network compromised. Furthermore, security lapses in controller-datapath communication can lead to illegitimate access and usage of network resources. On one hand, SDN enables applications to interact with the control plane to access network resources, deploy new functionalities and manipulate the behavior of the network. On the other hand, securing the network from malicious applications or abnormal behavior of applications is a serious security challenge in SDN. Network security is crucial for the success of a networking technology and communication networks must provide end-to-end communication security.

In this paper, we are describing security challenges in SDNs with proposed security solutions, and security platforms. An analysis of network security in SDN is presented in [5]. Scott *et al.* [5] provide a brief overview of security challenges in SDN and describe some of the existing frameworks for enhancing security in SDN. It is stated in [5] that even though security as an advantage of SDN has been recognized, still there are fewer SDN security solutions. However, [5] is limited in scope and does not cover the recent advances in SDN security. In this paper, we aim at providing a comprehensive and up-to-date overview of security in SDN by presenting security challenges and solutions related to individual SDN planes i.e., the application plane, control plane, and data plane. We also describe network-wide security solutions and security development platforms in SDN, categorize security solutions according to the ITU-T security recommendations, and briefly present costs of various security architectures.

This paper is organized as follows. Section II describes security in past programmable networking proposals followed by the concepts of SDN in Section III. Security challenges existing in each of the three planes (i.e., *Application, Control, and Data planes*) of SDN are presented in Section IV. Security proposals, platforms and solutions for each of the three planes in SDN are discussed in Section V. Network-wide security platforms for SDN are presented in Section VI. SDN-based virtual and cloud networks' security is described in Section VII. SDN security frameworks and platforms are categorized and discussed according to the ITU-T recommendations in Section VIII. Future research directions for security in SDN are outlined in Section IX and the paper is concluded in Section X.

II. Security in Programmable Networks: The Past

Security has been a daunting task in communication networks due to the underlying network complexities, proprietary and perimeter-based security solutions that are difficult to manage, and the weak notions of identity in IP networks. Similarly, the Internet architecture that defines procedures for usage of the underlying infrastructure [6], inherits the problems arising from the infrastructure, is ripe with security challenges and is stagnant to innovation. Therefore, many proposals have been put forward for (re)architecting the Internet to curtail its inherent limitations, and to minimize its complexities and security vulnerabilities. In this section, we discuss those proposals which either had an impact on network security or network security has been its important objective besides other objectives.

A. Active Networking

Active networking was proposed to enable programmability of nodes (e.g., routers and switches) through user injected programs [7]. In active networks, nodes perform customized computation on the payload that passes through them. Hence, the nodes can be tailored to function according to user or application requirements. The benefits of active networking are i) deployability of adaptive protocols, ii) capability to implement fine-grained application-specific functionalities in desired nodes within the network, and iii) user driven customization of the infrastructure to enable fast deployment of new services [8].

A major challenge for active networks was to secure active nodes from malicious user-injected programs. As a result, active security [9] and other security approaches [10] were proposed to ensure a node's security through authentication and authorization mechanisms. However, complexity in management and security of active nodes remained challenging tasks in active networks.

B. The 4D Approach

Greenberg *et al.* [11] associate the fragile nature of communication networks to the complex nature of control and management planes in traditional networks. It is illustrated that the lack of coordination between routing and security mechanisms result in a fragile network and security lapses. Henceforth, a clean-slate approach is proposed called the "*4D approach*," named after the four planes of decision, dissemination, discovery, and data. The 4D architecture completely re-factors the functionalities of a network and separate the network control from the forwarding substrate. The authors propose that a network architecture should be based on three key principles i.e., i) network-level objectives, ii) network-wide views, and iii) direct control [11].

In the 4D architecture, network security objectives are considered as network-level goals and network security is considered as an integral part of the network management. The separation logic was proposed to enable new, simpler, more robust, more reliable, and more secure control and management protocols from a centralized decision plane. The similarity in principle objectives (e.g., Data-Control plane separation) 4D and SDN shows that the SDN architecture is the recent version of the 4D architecture. Similarly, OpenFlow has sprung from the ideas of the 4D project as stated in [12].

C. SANE

Secure Architecture for the Networked Enterprise (SANE) [13] is a clean-slate protection architecture for enterprise networks. The design goals of SANE include an architecture that supports simple but powerful natural policies, independence from topology and network equipment, link layer security, protection of topology and services information from unauthorized access, and centralized definition and execution of all the policies [13]. The SANE architecture has a Domain Controller (DC) that performs three main functions. First, the DC authenticates users, hosts, and switches, and maintains a symmetric key with each for secure communication. Second, it advertises and controls access to available services. Third, the DC controls all the connectivity in a SANE network.

The idea of SANE emerged to solve the complexity of security systems (security boxes) used in enterprises. Traditionally, configuration of these boxes is complex, often dependent on network topology and based on addresses or physical ports that make network management difficult and result in fragile network security. SANE allows simple high-level policies which are expressed centrally and are enforced by a single fine-grained mechanism within the network. The complexity challenges are solved with the help of centralized decision making and reducing the number of trusted and configured components with simple and minimally-trusted forwarding elements. The SANE architecture has been extended to Ethane [14] that is based on the principle of incremental deployment in enterprise networks [15].

D. Ethane

Although, the main theme of the Ethane project [14] was to centralize the control logic of a network, it decouples the architecture from the infrastructure in a way that the forwarding switches are acting on behest of the centralized controller. The Ethane architecture comprises a centralized controller with global view of the network, simple and dumb Ethane switches with a simple flow table, and a secure channel to the controller. Besides proper policy management in the network, security has been considered as an integral part of the network management and hence identity-based access control has been considered for the architecture. Ethane follows the work of the SANE [13] architecture and has a resemblance to SANE. However, unlike SANE, in Ethane security is considered as a subset of network management, and due to its backwards-compatibility, it can be deployed incrementally.

Ethane is built around three main principles. First, the network is governed by policies which are declared over high-level names. Second, policies determine paths that packets follow. This makes it easy to control traffic and deploy new security services. For example, a policy might require that certain traffic should be forwarded to intrusion detection systems, security middle boxes, and firewalls etc. The third and most interesting principle is the strong binding between packets and their origin. The addresses used today are dynamic and change frequently. Hence, it is extremely difficult to reliably relate traffic to their source. In Ethane, policies declared over high-level names (e.g., users and hosts) and secure binding between packet headers and the physical entities that sent them enable tracking users and machines even if they move. This is a strong and distinct security feature of Ethane that must be incorporated in SDN architectures. Ethane is considered to be the predecessor of the current OpenFlow variant of SDN which can be seen in the architectural resemblance of both technologies.

III. Software Defined Networking Concepts and Implementation

SDN decouples the network control and forwarding functions. The control logic is separated from individual forwarding devices, such as routers and switches, and implemented in a logically centralized controller. Hence, the data plane becomes a set of simple forwarding devices which are managed by the

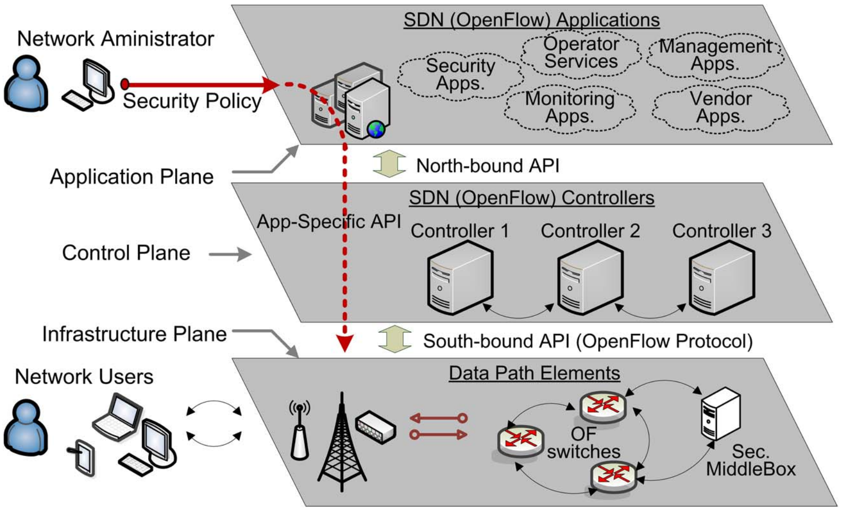

Fig. 2. SDN reference architecture with network constituents.

controller. The separation of control and data planes enables the network control to be programmable and the underlying infrastructure to be abstracted for applications and network services. In other words, SDN dis-aggregates traditional vertically integrated networking stacks to improve network feature velocity or to customize network operation for specialized environments.

In SDN, the controller is going to be talking and pushing instructions in real-time down to the network elements. Having the software layer above controlling the hardware underneath, applications can be built on top of the software layer to utilize and control heterogeneous network resources [16]. The aim of SDN is to allow network engineers and administrators to respond quickly to the changing business requirements. Network administrators can shape traffic from the central controller without having to touch the physical switches using the software to prioritize, redirect or block traffic either globally or in varying degrees down to individual packet levels [17].

The SDN reference architecture and the interaction between the SDN planes/layers is presented in Fig. 2. As shown in Fig. 2, a network administrator can set security policies for the network through the application plane and redirect network traffic to different security systems or middle-boxes using the control plane. Furthermore, the granularity of the security procedures can be extended to individual flows with the help of the SDN controller. Since OpenFlow [2] is considered to be the de-facto standard of SDN, most of the security systems and proposals are based on OpenFlow. Therefore, we describe the three planes of SDN in accordance with the OpenFlow architecture below.

A. *OpenFlow: Enabler of SDN*

The three tier approach of SDN has been implemented in the form of OpenFlow; a novel approach for networking with vendor-agnostic interface, e.g., the OpenFlow protocol between the control and data planes. OpenFlow [2] is a popular SDN technology describing how a logically centralized software controller and a network forwarding device should communicate, thus, making it possible to implement and deploy SDN technology in current networks. The OpenFlow controller provides a clear unified view of the network; making it easier to spot network vulnerabilities and intrusions, and implement security policies [18]. Security applications can be implemented on top of the control plane to deploy network security functions.

B. *Application Plane*

SDN enables applications to interact with and manipulate the behavior of network devices through the control layer. Applications benefit from the visibility of network resources in the controller and, therefore, can request the network states, and accessibility of network resources in specific ways. There is a requirement to couple applications with the underlying physical or virtual resources e.g., for topology and link discovery, firewall services, domain name services, network address translation services and deploying virtual private networks. Therefore, network operators and service providers desire to control, manipulate, manage and set policies by using applications for various network control, configurations and manipulation options [19].

In SDN, the control plane provides an abstract view and resource information of the entire network elements to SDN applications. In OpenFlow, the controller abstracts the network complexity, gathers network information through the south-bound API and maintains a logical map of the entire network. The network information can be provided to applications through the north-bound API of the controller. Hence, OpenFlow is a natural choice to implement network functions in the form of OpenFlow applications. Therefore, a variety of network security services are implemented on top of the OpenFlow controller as security applications which are described in latter sections.

C. *Control Plane*

In SDN, the control plane is taken out from individual network nodes and implemented in a separate logically centralized plane. The entity that implements the control plane functionalities is referred to as the SDN controller [20]. The SDN controller is responsible for managing and controlling the whole network through the NOS from a central ventage point with a global view of all the network resources. The NOS collects network information using APIs to observe and control a network conceptually much like a computer operating system. The operating system for networks provides a uniform and centralized programmatic interface to the entire network such that applications implemented on top of the operating system perform the actual management tasks [21].

Being an integral part of SDN, the controller is responsible for flow settings in data-path elements (e.g., OpenFlow switches) such that all the flow processing in the data-path is based on instructions from the controller. In the OpenFlow SDN architecture, the OpenFlow protocol provides an open and standard approach for the controller to communicate with switches [2]. When a packet of a new flow arrives in the switch, the switch checks its flow table for flow rules corresponding to that particular packet. If a matching entry for the flow exists, instructions for that specific flow are executed, otherwise the flows are forwarded to the controller. The controller then sets the flow rules in the switch flow tables to either forward the flow packets to a particular port or drop packets coming from that particular source.

The initial design of OpenFlow considered a single OpenFlow controller for simplicity. Recent OpenFlow architectures support multiple controllers which can be distributed in the network to achieve higher scalability and availability.

D. Data Plane

SDNs' separation of control and data planes refers to making the forwarding devices simple and remotely controllable via open interfaces. Forwarding devices such as routers, switches, virtual switches and access points can be (re)configured and (re)programmed for different purposes including traffic isolation and virtualization via a remote procedure call from the controller using a secure communication channel. The most suitable example of SDN forwarding devices is the OpenFlow switch. OpenFlow switches are simple and, thus, future-proof since forwarding policies are imposed by the controller software rather than by the switch hardware or firmware. Any router or Ethernet switch having flow tables can be programmed to be an OpenFlow switch using the OpenFlow protocol. When programmed to an OpenFlow switch, the flow tables of the switch or router are used to maintain flow entries with an associated set of actions for each flow [2].

The OpenFlow switch can have an extensible set of functionalities but the minimum requirements are that it must contain; i) a flow table with actions associated with each flow for processing the flows accordingly; ii) a secure channel to the controller to allow communication of instructions and packets; and iii) an OpenFlow protocol providing an open and standard mechanism for the controller to communicate with the switch. An OpenFlow switch must be capable of forwarding the packets according to the flow rules installed in the flow tables. There can be dedicated OpenFlow switches or general purpose Ethernet switches or routers enabled with OpenFlow. The prior does not support normal Layer 2 and Layer 3 processing, whereas the latter have OpenFlow protocol and interfaces added as new features [2].

E. Standardization Activities and Industry Acceptance

SDN has received overwhelming attention from the industry and academia. This necessitated fast standardization leading to initiating activities in Standard Development Organizations (SDOs), industry and community consortia. These standardization bodies deliver results that are considered as the *de facto* standards which mostly come in the form of open source implementations [22]. The Open Networking Foundation (ONF) has been perceived as the leader for SDN standardization that promotes the adoption of SDN through the development of the OpenFlow protocol as an open standard for controller-data path communication.

ONF is structured in many technical working groups (WGs) for architecture and framework, extensibility, configuration and management, forwarding abstractions, testing and interoperability. The Internet Engineering Task Force (IETF) has created the SDN Research Group (SDNRG) that focuses on research aspects for the evolution of the Internet. IETF has published Internet Drafts on security requirements in SDN [23], security of OpenFlow switch [24], and SDN and NFV security architecture [25]. The International Telecommunication Union's Telecommunication sector (ITU-T) has started Study Groups (SGs) to develop recommendations for SDN, and a Joint Coordination Activity on SDN (JCA-SDN) to coordinate the standardization work. We have presented SDN security according to the ITU-T security recommendations [26] in Section VIII.

The networking industry has opted for SDN as a new paradigm and a revolution in networking technologies. In March 2011 Deutsche Telekom, Facebook, Google, Microsoft, Verizon and Yahoo! formed the Open Networking Foundation (ONF) to promote SDN technologies [18]. Adopting the SDN technology, Google has deployed OpenFlow in one of their backbone networks [27], Cisco has developed SDN-based Cisco Application Centric Infrastructure [28], VMware has developed network virtualization and a security platform, i.e., NSX [29] for a software-defined data center, and many other companies are already manufacturing SDN products as listed in [30]. According to the SDN and Network Function Virtualization (NFV) community, i.e., the SDNCentral [31], the SDN market is expected to surpass $35 Billion by 2018. One can deduce from these facts that the future of networking lies in SDN.

IV. Security Challenges in SDN

Separation of the planes and aggregating the control plane functionality to a centralized system (e.g., OpenFlow controller) can be fundamental to future networks; however, it also opens new security challenges. For example, communication channels between isolated planes can be targeted to masquerade one plane for attacking the other. The control plane is more attractive to security attacks, and especially to DoS and DDoS attacks, because of its visible nature. The SDN controller can become a single point of failure and render the whole network down in case of a security compromise. Network resource visibility is of paramount importance in SDN, but these resources must not be visible to all or unconcerned applications.

The list of security challenges in SDN is expected to grow with the gradual deployment of SDN technologies. In order to

TABLE I
MAJOR SECURITY THREATS IN THE SDN PLANES

SDN Plane or Layer	Type of Threat	Description
Application Plane	Lack of authentication & authorization	No compelling authentication & authorization mechanisms for applications and more threatening in case of large number of third-party applications.
	Fraudulent flow rules insertion	Malicious or compromised applications can generate false flow rules and it is difficult to check if an application is compromised.
	Lack of access control & accountability	Difficult to implement access control & accountability on third-party applications and nested applications that consume network resources.
Control Plane	DoS attacks	Visible nature, centralized intelligence and limited resource of the control plane are the main reasons for attracting DoS attacks.
	Unauthorized controller access	No compelling mechanisms for enforcing access control on applications.
	Scalability & availability	Centralizing intelligence in one entity will most likely have scalability and availability challenges.
Data Plane	Fraudulent flow rules	Data plane is dumb and hence more susceptible to fraudulent flow rules.
	Flooding attacks	Flow tables of OpenFlow switches can store a finite or limited number of flow rules.
	Controller hijacking or compromise	Data Plane is solely dependent on the control plane that makes the data plane security dependent on controller security.
	TCP-Level attacks	TLS is susceptible to TCP-level attacks.
	Man-in-the middle attack	Due to optional use of TLS, and complexity in configuration of TLS.

take full advantage of SDN, these challenges must be highlighted so that proper security measures can be taken proactively. Therefore, security challenges and threats existing in SDN are discussed in this section. From a basic point of view, security vulnerabilities in SDNs are concentrated around the main areas of i) applications, ii) control plane, and iii) data plane. Hence, security challenges existing in the three planes(or SDN layers) are described below. The major and most common security challenges are presented in Table I.

A. Application Plane Security Challenges

SDN has two principle properties which make the foundation of networking innovation on one hand, and the basis of security challenges on the other. First, the ability to control a network by software, and second, centralization of network intelligence in network controllers [32]. Since most of the network functions can be implemented as SDN applications, malicious applications if not stopped early enough, can spread havoc across a network. Therefore, we describe security challenges related to SDN applications in this section.

Since there are no standards or open specifications to facilitate open APIs for applications to control network services and functions through the control plane [19], applications can pose serious security threats to network resources, services and functions. Although OpenFlow enables deploying flow-based security detection algorithms in the form of security applications, there are no compelling OpenFlow security applications [33]. Besides that, there are no agreed-upon compelling development environments, and network programming models or paradigms. The variety and multitude of vendor and third-party applications developed in different independent development environments using different programming models and paradigms could create interoperability limitations and security policy collision. Some of the threatening security challenges posed by SDN applications are described below.

1) Authentication and Authorization: Authenticating applications in the current fast trends of software engineering and emerging feats of hacks will be a major issue in the SDN domains. In OpenFlow, applications running on the controller implement a majority of the functionalities of the control plane and are typically developed by other parties than the controller vendors. These applications inherit the privileges of access to network resources, and network behavior manipulation mostly without proper security mechanisms for protecting network resources from malicious activities [34]. Hence, authentication of the increasing number of applications in programmable networks with centralized control architecture is a major security challenge.

Kreutz *et al.* [32] presented threat vectors to describe security vulnerabilities in SDN. It is described that there are no compelling mechanisms to establish a trust relationship between the controller and applications in SDNs. Hence, a malicious application can potentially create havoc in the network since the SDN controllers provide abstractions that are translated to configuration commands for the underlying infrastructure by applications. Similarly, if an application server that stores the details of users is compromised, credentials of legitimate users can be used to inject authorized, but, forged flows into the network. Moreover, various techniques exist to certify network devices in a network, but there are no mechanisms to certify network applications. Since network functionality in SDN is implemented in applications, a centralized system to certify SDN applications is required but it is not available yet.

2) Access Control and Accountability: Since applications implement most of the network services in SDN, proper access control and accountability mechanisms are needed to ensure the security of a network. To understand the possible security threats related to access control and accountability in SDN, consider the following example applications.

Hartman *et al.* [35] identify three classes of applications that can affect network security in SDN. First, network sensitive applications that require particular network characteristics such as path characteristics, cost of traffic flows, etc. Second, applications that provide services for the network such as access control or firewall, and content inspection or intrusion detection services. Thirdly, packaged network services that combine applications from the first and second classes, or applications requesting instantiation of another application as a virtual element in the network. Hence, a malicious application can bypass access control by using an instance of the second class application.

Besides that, access control and accountability of nested applications (e.g., an application using an instance of another application) will be a real challenge in SDN. An example of such applications is mentioned in [20] stating that applications in SDNs can be either SDN-aware or unaware of SDNs. SDN-aware applications are capable of locating and directly communicating with the SDN controllers while the unaware SDN applications communicate indirectly with application datagrams in specific formats [20]. In the latter case, a legitimate, but compromised SDN application can become a gateway for unauthorized access to the network control plane. Similarly, maintaining accountability for usage of network resources by nested applications is another challenge.

B. Control Plane Security Challenges

In SDN, the control plane (e.g., OpenFlow controller) is a centralized decision-making entity. Hence, the controller can be highly targeted for compromising the network or carrying out malicious activities in the network due to its pivotal role. The main security challenges and threats existing in the control plane are described below.

1) Threats From Applications: Applications implemented on top of the control plane can pose serious security threats to the control plane. Generally, the controller security is a challenge from the perspectives of controller capability to *authenticate* applications, and *authorize* resources used by applications with proper isolation, auditing, and tracking [35]. Added to that, there is a need of separating different applications according to their security implications before access to network information and resources is provided. The separation of applications is highly important to distinctly authenticate and authorize third-party and operator applications having established auditing for each application.

Different applications having different functional requirements from the underlying controller and data path must qualify different security requirements. Some applications, e.g., load-balancing application might need network statistics such as bytes or packets counter values from the switch to perform load balancing. Other applications, e.g., intrusion detection application might need to inspect packet header fields. Besides that, vendor and third-party applications must have different privileges to access network information and resources. For example, *participatory networking* in SDN is presented in [36] to enable end-users and their applications to take part in network configuration. These kinds of user-applications must be scrutinized properly before access to the network is provided having comparatively less privileges than vendor applications. Therefore, a customized security enforcement mechanism for various types of applications is required in the north-bound API of the controller. Such customized security procedures based on the type or categories of applications have not been demonstrated yet.

2) Threats Due to Scalability: In OpenFlow, most of the complexity is pushed towards a controller where forwarding decisions are taken in a logically centralized manner [37]. If the controllers are required to install flow rules for each new flow in the data path, the controller can easily become a bottleneck. The authors in [38] analyzed that todays controller implementations are not capable to handle the huge number of new flows when using OpenFlow in high-speed networks with 10 Gbps links. It is described in [39] that the lack of scalability enables targeted attacks to cause control plane saturation that has more detrimental results in SDNs than traditional networks. Hence, controller scalability makes it a favorite choice for DoS and distributed DoS attacks.

Another challenge for the currently available controller implementations is specifying the number of forwarding devices to be managed by a single controller to cope with the delay constraints. If the number of flows on the controller increase, there is a high probability that the sojourn time will increase which is deeply dependent on the processing power of the controller. This limitation of controllers' capabilities can lead to single point of failure. To avoid the challenges of a controller being a single point of failure, the use of multiple controllers is suggested. However, it is demonstrated in [40] that simply utilizing multiple controllers in SDNs cannot protect the network from single point of failure. The reason is that the load of controllers carrying the load of the failed controller can exceed their capacity and hence will result in a worse situation, e.g., cascading failures of controllers [40].

3) DoS Attacks: DoS and distributed DoS attacks are the most threatening security challenges for the SDN controller. DoS attack is an attempt to make a (network) resource unavailable to legitimate users. A DoS attack on SDN is demonstrated in [41] that exploits the control-data planes separation logic of SDN. A network scanning tool is developed that can identify an SDN network with the help of flow response times. Since for each new flow, the data path queries the controller, there is a difference in flow response times for new and existing flows. The scanner gathers the time values with the help of header field change scanning, which scans networks as changing network header fields. Having found the network to be SDN, specifically crafted flow requests are transmitted to the target network which are forwarded by the datapath to the controller. Increasing the number of flows in the datapath will make the switches bombard flow setup requests on the controller and hence eventually cause it to break.

A DoS attack on the SDN controller is also demonstrated in [42] where an attacker continuously sends IP packets with random headers to put the controller in non-responsive state. The authors [42] use a secondary controller for improved resilience, however, a DoS or DDoS detection mechanism is still required since the secondary controller can also be susceptible to DoS or DDoS attacks. Hence, the use of multiple controllers is not the answer to DDoS attacks since, it can lead to cascading failure of multiple controllers as demonstrated in [40].

4) Challenges in Distributed Control Plane: To manage a huge number and variety of devices that cannot be managed by a single SDN controller, multiple controllers have to be deployed which divide the network into different sub-domains. But, if the network is divided into multiple software-defined sub-networks, information aggregation and maintaining

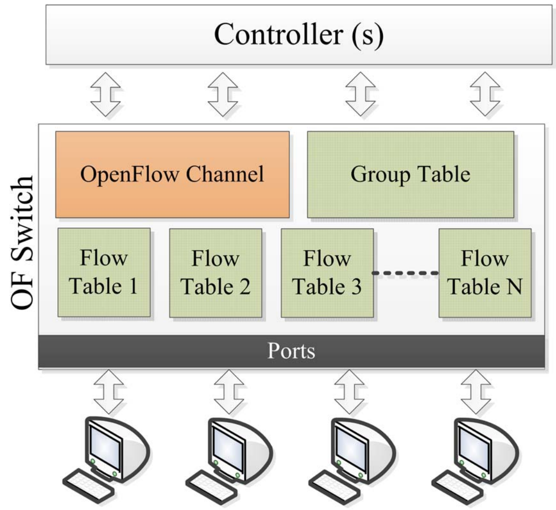

Fig. 3. OpenFlow switch architecture.

different privacy rules in each sub-network will be a challenge. This can be described with the example of the Application-Layer Traffic Optimization (ALTO) [43] application implemented in the SDN application plane. The ALTO application needs network level information such as topology and link information from the controller to optimize traffic for a specific application. In [19], the authors discuss a model where managed service providers (MSPs) provide network services to applications. In this case, an application traversing multiple network domains may pose serious security threats related to *authorization*, *authentication* and *privacy* by acquiring network resource information of third party networks without having proper service-level agreements (SLAs) in place.

5) Other Challenges: Since the controller is responsible for network-wide policy enforcement, the controller-switch semantic gap, distribution of *access control* supporting aggregated flows, multi-tenant controllers, and multiple controllers in a single domain can create configuration conflicts [44]. In multiple OpenFlow infrastructures, inconsistency in the controller configurations will result in potential inter-federated conflicts [44]. For example, if the state of the network changes, all the controllers might not receive the network information at the same time. Hence, some stateful applications such as firewalls might behave incorrectly due to the controller non-visibility into past events.

C. Data Plane Security Challenges

In OpenFlow networks, the OpenFlow controller installs flow rules in the OpenFlow switch's flow tables. These flow rules can be installed before a new host sends packets (proactive rule installation) or upon the first packet from a new host (reactive rule installation). As shown in Fig. 3, a switch has a limited number of flow tables where flow rules are installed according to the controller's view of the network. Since, the decision making capability has been taken out of switches, the first and foremost security challenge is recognizing genuine flow rules and differentiating them from false or malicious rules. The second challenge is based on the number of flow entries a switch can maintain. In OpenFlow, a switch has to buffer flows until the controller issues flow rules. This makes the data plane prone to saturation attacks, since it has limited resources to buffer unsolicited (TCP/UDP) flows.

In SDN, the security of the control plane has direct implications on the data plane as described in [39]. It means that if a controller is compromised, the whole network comprising a variety of data plane nodes will be compromised. In split architectures such as SDN, if a switch does not receive forwarding instructions from the control plane, either because of control plane failure or control plane disconnection, the data plane becomes practically offline [45]. Hence, the switch-controller link can be a favorable choice for attacking the network. The control and data planes separation can enable an attacker to manipulate flows stealthily through the manipulation of OpenFlow rules, resulting in various active network attacks, such as man-in-the-middle attack, and black-hole attacks.

The original OpenFlow specification defines Transport Layer Security (TLS) [46] and Datagram Transport Layer Security (DTLS) [47] for the controller-switch communication. However, the latter versions of OpenFlow (e.g., OpenFlow v1.3.0) makes the use of TLS optional. Moreover, it is described in [48] that TLS has a higher technical barrier for operators due to its complexity of configuration. The configuration steps include generating a site-wide certificate, generating controller and switch certificates, signing the certificates with the site-wide private key, and installing correct keys and certificates on all of the devices. Therefore, many vendors have skipped support for TLS in their OpenFlow switches.

Complexity in configuration and optional use of TLS can make the control channel vulnerable to various types of attacks. Benton *et al.* [48] describe how man-in-the-middle attacks are more lethal in OpenFlow networks than traditional networks due to constant connectivity and lack of authentication in the plain-text OpenFlow TCP control channel. As a result, an attacker can immediately seize full control of any down-stream switches and execute fine-grained eavesdropping attacks [48]. Added to that, it is demonstrated in [49] that the use of TLS does not provide any TCP-level protection and as such, can be prone to TCP-level attacks.

Furthermore, SDN can allow the network traffic to be routed through a centralized firewall to secure the data-plane. However, monitoring messages between a switch and a controller can take long enough time that the switch resources are exhausted by false flows leading to flooding or a DoS attack compromise. Dimitri *et al.* presented in [50] that it is hard to achieve carrier grade requirement of restoration within 50ms in large OpenFlow networks. Delays in setting up the flow rules can introduce difficulties in authorization and authentication, and inspection of traffic contents and flow rules by security applications.

V. Security Solutions of SDN Planes

In SDN, the control plane is logically centralized to make centralized decisions based on the global view of the network. As a result, logically the SDN architecture supports highly reactive security monitoring, analysis and response systems to facilitate network forensics, security policy alteration and security

TABLE II
SECURITY PLATFORMS FOR THE SDN PLANES OR LAYERS

SDN Plane or Layer	Security Solution	Targeted Threat	Solution Type
Application Plane	FRESCO [52]	Threats within/from apps.	Security apps. development framework
	PermOF [34]	Access control	Apps. permission system
	Assertion [53]	Flow rules contradiction	Apps. debugging framework
	Flover [54]	Security policy violation	Security policy verification apps.
	OFTesting [55]	Faulty OF programs	Apps. testing framework
Control Plane	SE-Floodlight [56]	Apps. authorization	Secure controller arch. & secure App-Ctrl API
	HybridCtrl [57]	Controller scalability	Hybrid (reactive/proactive) controller architecture
	DISCO [58], [59]	Controller scalability	Distributed controller architecture
	Ctrl-Placement [60], [61], [62]	Controller availability	Controller placement frameworks
	HyperFlow [63]	Controller availability	Distributed control plane
	DDoSDetection [64]	DDoS attack	Detection framework
Data Plane	FortNOX [65]	Flow rules contradictions	Controller framework
	FlowChecker [44]	Faulty flow rules	Configuration verification tool
	VeriFlow [66]	Faulty flow rules	Network debugging tool
	Resonance [67]	Access control	Access control & policy enforcement framework
	CPRecovery [42]	Controller availability	Controller replication framework

service insertion [51]. For network forensics SDN facilitates quick and adaptive threat identification through a cycle of harvesting intelligence from the network to analyze, update the policy and reprogram the network accordingly. SDNs facilitate dynamic security policy alteration to define a security policy, push it to the network elements and reduce the chances of misconfiguration and policy conflicts across the network in run time. Because of the global network visibility, security services such as firewalls and intrusion detection systems (IDS) can be easily deployed on specified traffic according to defined security policies.

In this section, security measures, proposals and platforms for securing application, control and data planes are discussed. Security solutions that explicitly strengthen the security of each plane are presented in Table II. The organization of solutions with respect to SDN planes in Table II shows its highest impact in terms of security on that particular plane.

A. Application Plane Security Solutions

In SDN, the controller acts as an intermediate layer between the network hardware and applications and hides the network complexity from applications. Hence, the centralized control architecture enabled by SDN makes it easy to deploy new applications that would retrieve network statistics and packet characteristics through the controller to implement new security services. Therefore, various network programming languages, such as Frentic [68], Procera [69], and NetCore [70] are proposed that simplify the development of applications in SDN. Furthermore, FRESCO [52] is proposed to enable development of OpenFlow security applications.

The FRESCO scripting language enables developers to implement new security applications which can be deployed on any OpenFlow controller or switch implementation. Similarly, various security frameworks are proposed to check whether SDN applications comply with network security policies or not. These proposals and frameworks are described below.

1) Access and Permission Control: Applications need to work in its functional boundaries and have controlled access to network resources. PermOF [34] is a fine-grained permission system used to provide controlled access of OpenFlow controller and data path to OpenFlow applications. The design is based on a set of permissions and isolation mechanisms to enforce the permission control. The permission set is categorized into read, notification, write, and system permissions. The permission set is further divided into sub-categories as shown in Table III.

TABLE III
PERMISION SET OF THE PERMOF [34]

Category	Permissions
Read	read_topology read_all_flow read_statistics read_pkt_in_payload
Notification	pkt_in_event flow_removed_event error_event topology_event
Write	flow_mod_route flow_mod_drop flow_mod_modify_hdr modify_all_flows set_device_config set_flow_priority
System	network_access file_system_access process_runtime_access

The read permission is used to manage the availability of sensitive information to an application. The notification permission is used to manage whether an app should be notified of certain events in real time. The write permissions manage the ability of applications to modify states in the controller or switches, and the system permissions manage an application's access to local resources provided by an OS. The isolation framework maintains the controller superiority over applications, isolates control flow and data flow, and enables the controller to mediate all the applications' activities with the outside world. However, there is no experimental evaluation of the proposed framework.

2) Compliance With Network Security: In SDN, applications must have a consistent view of the network and be aware of the changing network conditions. A method for verifying and debugging SDN applications to stay aware and consistent with the changing network conditions is presented in [53]. Assertion-based debugging and verification language is developed to enable application developers to verify dynamic properties of controller application via high-level program statements. Assertion-based methods help to catch bugs in programs before they are deployed. Since, VeriFlow [66] provides mechanisms to inspect flow rules in run-time, the proposed verification procedure of [53] uses the VeriFlow verification algorithm with an incremental data structure to efficiently verify properties with dynamically changing verification conditions.

Flover [54], a model checking system for OpenFlow verifies that the aggregate of flow policies does not violate the network's security policies. Flover is implemented as an OpenFlow application running on the controller to check that new flow rules created by the controller are consistent with a set of specified properties. There are other proposals such as [55], where the authors propose an automatic testing procedure for identifying bugs in OpenFlow programs. The ndb framework [71] provides a debugging tool for network programmers to find the root cause of bugs in a network. The OFRewind [72] can record and replay selected traffic to trace network anomalies. *ndb* and *OFRewind* frameworks can be used to trace those applications that induce security threats in a network.

3) Remarks: The platforms and proposals discussed above help in developing security applications and providing security to the control plane from malicious applications. It is important to mention that there is very little effort to strengthen the security of applications' data and the applications themselves. Besides that, there are no mechanisms for differentiating between third-party or user applications and operator or network service applications. Moreover, access control and accountability procedures for nested applications have not been demonstrated yet.

B. Control Plane Security Solutions

The control plane security solutions are categorized into proposals and approaches for securing the control plane from I) malicious or faulty applications, II) circumventing SDN security by targeting the scalability of the control plane, III) Dos or DDoS attacks, and IV) ensuring the control plane security and availability through reliable controller placement.

1) Applications: Since applications access network resources and information through the control plane, it is very important to secure the control plane from malicious or faulty applications. Moreover, the control plane must ensure access to legitimate applications according to their functional requirements but within the security constraints.

Security-enhanced (SE) Floodlight controller [56], an extended version of the original floodlight controller [73], is an attempt towards an ideal secure SDN control layer. The SE-Floodlight controller provides mechanisms for privilege separation by adding a secure programmable north-bound API to the controller to operate as a mediator between applications and data plane. It introduces a run-time OpenFlow application verification module for validating the integrity of class modules that produce flow rules. For role-based conflict resolution, SE-Floodlight assigns authorization roles to OpenFlow applications to resolve rule conflicts by comparing the authoritative roles of producers of conflicting rules. Similarly, it can restrict PACKET_OUT messages produced by various applications and hence secure flow rule mediation. The SE-Floodlight controller also introduces a new OpenFlow audit subsystem that can track all security-related events occurring within the OpenFlow control layer.

2) Controller Scalability: In the OpenFlow standard of SDN, a controller installs separate rules for each client connection, also called "microflow," leading to installation of a huge number of flows in the switches and a heavy load on the controller. Therefore, various approaches are suggested to either minimize the load on a controller, distribute control plane functionalities, or maximize the processing power and memory of controllers. Besides that, OpenFlow supports the use of wildcards so that the controller directs an aggregate of client requests to server replicas. The wildcard mechanisms exploit the switch support for wildcard rules to achieve higher scalability besides maintaining a balanced load on the controller. These techniques use algorithms that compute concise wildcard rules that achieve target distribution of the traffic and automatically adjust to changes in load balancing policies without disturbing existing connections.

A comparative analysis of various reactive and proactive OpenFlow controller paradigms for scalability is presented in [57]. Reactive controllers receive the first packet of flow from the switch to populate the flow table in the switch for that particular flow, whereas proactive controllers set flow rules before the flows arrive at the switch based on some pre-defined forwarding rules. The paper [57] demonstrates that proactive controllers are more scalable than its counterpart. However, a pure proactive controller would need to know all the traffic flows in advance which is not practically possible. Therefore, the author suggests a hybrid controller architecture in which the controllers act reactively to configure routes and have some intelligence to act proactively to understand the traffic behavior and define a path in advance.

There are efforts to increase the processing power of the controllers and share responsibilities among a set of controllers. McNettle [74] is an extensible SDN controller with multiple CPU cores developed to scale and support control algorithms. It requires global visibility of state changes occurring at flow arrival rates. Programmers can extend McNettle with a high-level functional programming language. Compared to NOX, McNettle is more scalable as it can scale up to 46 cores while NOX can scale up to 10 cores besides being more efficient performance-wise. In [12] and [74] parallelism through multicore processors is proposed to increase the processing performance of the controllers for higher scalability and availability.

The distributed SDN control plane (DISCO) is presented in [58], [59] to provide control plane functionalities to distributed, heterogeneous, and overlay networks. DISCO is implemented on top of the Floodlight [73] OpenFlow controller and uses Advanced Messaging Queuing Protocol (AMPQ) [75]. It is composed of two parts i.e., intra-domain and inter-domain. The intra-domain modules enable network monitoring and manage flow prioritization for the controller to compute paths of priority flows. These modules help in dynamically reacting to network issues by redirecting and/or stopping network traffic according to the criticality of flows. The inter-domain part manages communication among controllers and is composed of a messenger and agents. The messenger module discovers neighboring controllers and provides a control channel between those neighboring domains. The agents use the channels provided by the messenger to exchange network-wide information with other controllers.

HyperFlow [63] is a physically distributed and logically centralized event-based scalable control platform. HyperFlow allows network operators to deploy multiple controllers, being capable of local decision making, in order to maximize controller scalability and minimize flow-setup time. Heller *et al.* [60] provide a trade-off between availability, state distribution and suggest to place controllers to minimize latency as a starting point, and then use load balancing algorithms to balance the load among the controllers. Load balancing techniques described in [76], [77] can be used to increase the scalability of the SDN control plane.

3) DoS Mitigation: DoS or DDoS attacks can be mitigated by analyzing flow behavior and flow statistics stored in OpenFlow switches. Since the switch statistics can be easily fetched in the OpenFlow controller, the process of statistics collection in OpenFlow is comparatively cost effective due to low overhead. A lightweight DDoS flooding attack detection using Self-Organizing Maps (SOM) [78] is presented in [64]. SOM is an artificial neural network used to transform a given n-dimensional pattern of data into a 1- or 2 dimensional map. The transformation process carries out topological ordering, where patterns of data with similar statistical features are gathered for further processing. In [64], the mechanism of SOM is used to find hidden relations among flows entering the network.

The DDoS detection method in [64] uses three modules i.e., flow collector, feature extractor, and classifier. The flow collector module gathers flow entries from all the Flow Tables of OpenFlow switches during predetermined intervals. The feature extractor module extracts features that are important for DDoS attacks, gathers them in 6-tuples, and passes them to the classifier. The extracted features include, average of packets per flow, average of Bytes per flow, average of duration per flow, percentage of pair-flows, growth of single-flows, and growth of different ports. The classifier analyzes whether a given 6-tuple corresponds to an attack or normal traffic using SOM. SOM is trained with a sufficiently large set of 6-tuple samples collected either during an attack or normal traffic to create a topological map where different regions present each type of traffic. Henceforth, whenever the trained SOM is stimulated with a 6-tuple extracted from collected flow entries from OpenFlow switches, it will be able to classify a traffic either as normal traffic or an attack.

4) Reliable Controller Placement: The controller placement problem is presented in [60], where it is shown that the number of controllers and topological locations of controllers are two key challenges for network scalability and resilience in SDN. Therefore, optimal placement of the controller has garnered much attention of the research community and a number of algorithms for optimal placement have been examined and tested. The Simulated Annealing (SA) algorithm, a generic probabilistic algorithm, has been favored as the most optimal algorithm for controller placement in [61], [62], and [79]. For improving network resilience through efficient controller placement, a minimum-cut based graph partitioning algorithm is proposed in [45].

The controller placement problem with its NP hardness is described in [61] in order to maximize the reliability of SDN control operations while meeting the response-time requirements. The authors suggest that state synchronization and control coordination between controllers is necessary and can be achieved with techniques such as controller's hierarchy. For optimal placement, the expected percentage of control path loss is used as a reliability metric, where the control path loss is considered as the number of broken control paths due to network failures. The proposed mechanisms in [61] optimize the network by minimizing the expected percentage of control path loss. Several algorithms with their benefits for controller placement are examined using real topologies, and tradeoffs between reliability and latencies are presented.

Dynamic Controller Provisioning Problem (DCPP) has been addressed in [79]. The authors propose a framework for dynamically deploying multiple controllers in WAN in which both the number and locations of controllers are adjusted according to network dynamics. DCPP has been formulated as an Integer Linear Program (IGP) that uses traffic patterns to minimize the costs of switch state collection, inter-controller synchronization and switch-to-controller reassignment. Moreover, the controller deployment strategy described in [79] provides a fair trade-off between flow-setup time and communication overhead in SDNs.

Pareto-based Optimal Controller-placement (POCO) framework is presented in [80] to enhance resilience in SDN. The authors suggest that a single controller might be enough to satisfy latency constraints, however, many more controllers (at least 20% of all nodes need to be controllers) are necessary to meet the requirements of network resilience. The proposed framework in [80] optimizes networks with respect to inter-controller latency, load balancing between controllers, and trade-off considerations between latency and failure resilience.

5) Control-Data Plane Intelligence Tradeoff: Intelligence tradeoff between the controller and switches can be used to increase the controller availability. Devolved OpenFlow or DevoFlow [81] is one such approach that modifies the OpenFlow model in order to minimize control-data planes interaction. The architecture devolves some control back to the switches while maintaining central control of the controller. The authors suggest that central visibility of all the flows might not be necessary but rather costly in terms of scalability. Hence, DevoFlow is designed such that it uses wild-carded OpenFlow rules, and the switches can take local routing decisions where per-flow vetting by the controller might not be necessary. Most of the microflows in DevoFlow are handled in the data plane, however, operators can manage or scrutinize any flow that matter to them for management purposes.

In OpenFlow, a controller obtains the network topology information using the Link Layer Discovery Protocol (LLDP) to get the connection status of OpenFlow switches including the switch and the port connection information [82]. For fault management, the controller can use LLDP to monitor links in the network. In this case, the controller is required to be involved in all the LLDP monitoring messages which can result in scalability limitation. To overcome this limitation, an architecture is proposed in [83] in order to offload the link monitoring capability of Operation Administration and Maintenance (OAM) from the controller to the switch. The platform [83] proposes a general message generator and processing function in the switches, and extension in OpenFlow 1.1 protocol to support the monitoring function.

The Network Core Programming Language (NetCore) [70] is a high-level, declarative language for expressing packet-forwarding policies in SDNs. NetCore proposes novel compilation algorithms and a run-time system to conciliate the delay incurred by involving a controller in all packet processing decisions. NetCore divides packet-processing responsibilities among controllers and switches with the help of compilation algorithms coupled with the run-time system.

C. Data Plane Security Solutions

As a starting point, the data plane must be secured from malicious applications which can install, change or modify flow rules in the data path. Hence, fine-grained security enforcement mechanisms such as authentication and authorization are used for applications which can change the flow rules. FortNox [65] is one such platform that enables the NOX OpenFlow controller to check flow rule contradictions in real-time and authorize OpenFlow applications before they can change the flow rules.

FortNOX provides role-based authorization through digital signatures and security constraint enforcement through a software extension in the NOX OpenFlow controller. Using a live rule conflict detection engine, FortNOX mediates all OpenFlow rule insertion requests with the help of a rule conflict analysis algorithm i.e., "*alias set rule reduction*." Once a flow rule is inserted by a security application, FortNOX restricts other applications to insert contradicting flow rules in the same OpenFlow network.

FlowChecker [44] is a configuration verification tool used to identify inconsistencies in OpenFlow rules within a single switch or multiple inter-federated datapath elements. FlowChecker can be used as an OpenFlow application or a master controller to validate, analyze, and enforce OpenFlow end-to-end configurations at run-time. VeriFlow [66] is a network debugging tool used to find faulty rules inserted by SDN applications and prevent them from causing anomalous network behavior.

Since controller connectivity is vital for the functionality of a switch, redundant connections or fast link recovery mechanisms must be in place for communication between the two. The OpenFlow protocol itself helps in recovery of the switch by using connection detection techniques to check connection to the controller, such as sending activity probe messages periodically to the controller. OpenFlow protocol also provides the flexibility to configure a secondary connection with a backup controller to use if the first controller fails. In [42] controller replication is proposed to maintain the switch operation even if the main controller fails. In this case, a switch periodically sends probe messages to the controller. If the controller does not reply in a specific time interval, the switch assumes that the controller is down. Then, the OpenFlow switch tries to connect with the secondary controller by performing a handshake and establishing a connection immediately [42].

Proper network planning and segmentation can also help to strengthen the resilience of OpenFlow switches and maximize their connectivity with controllers. An OpenFlow switch which is always connected to the controller will be less prone to saturation attacks, due to the fact that it will not be required to store unsolicited flows for longer duration. It is demonstrated in [45] that the length of the path between a switch and a controller is directly proportional to connectivity loss. Therefore, it is suggested in [45] that, besides the optimal number of switches under a controller, the length of paths between controllers and switches must be optimally short. This will not only improve the system's performance in terms of delay constraints, but enable fast restoration, improve content availability to security applications, and enable fast security analysis.

VI. Network-Wide Security in SDN

Network programmability brought about by SDN has enabled the deployment of network security services, altering security policy and performing network forensics at run time. An interesting capability of SDN is that the route a packet takes can be easily determined, since flow management and flow forwarding decisions are taken by the centralized control plane. Hence, due to centralized control architecture and global visibility of the network state, mechanisms such as packet traceback [71] can be used to easily trace malicious users or hosts and efficiently stop/counter malicious activities. Therefore, security service insertion in SDN is cost effective, easy and more deterministic unlike traditional networks where security services must be distributed at different entry points without knowing the paths that packets follow.

The benefits of centralized control for secure cloud computing is presented in [85] while the ease of deploying new security services is demonstrated in [86] showing the implementation of MPLS Virtual Private Networks (MPLS VPNS) in OpenFlow networks. The use of SDN for improving anomaly detection systems in small and home networks is demonstrated in [87]. OpenSafe [88] using ALARMS (A Language for Arbitrary Route Management for Security) to manage the routing of traffic through network monitoring devices is an example of economical deployment of security monitoring systems in SDN. Software-defIned Middlebox PoLicy Enforcement (SIMPLE) [89] is another example of deploying security middle-boxes without modification in middle-boxes or SDN architectures. A multi-tier security architecture for future mobile networks leveraging the SDN centralized network control is presented in [90].

In the previous section (Section V) we described security solutions that secure a particular SDN plane or layer. In this section, we describe various security approaches such as flow sampling, contents inspection, security middle boxes, etc. that can secure the whole network through merging the capabilities of the application, control and data planes. Table IV presents various security solutions that secure various parts of the network, e.g., SDN plane (SDN Layer) and_or interface(s). Below, we start from describing the security system development platforms in SDN, the basis of enabling security such as flow sampling, content inspection, etc. and complete the section with secure networking architectures in SDN.

TABLE IV
SDN SECURITY PLATFORMS

Security Solutions	**Security Type**	**Target Plane**			**Interface**	
		App.	Ctrl.	Data	App-Ctrl	Ctrl-Data
FRESCO [52]	Anomaly detection and mitigation framework		✓		✓	
PermOF [34]	Permission control system for OF Apps.		✓	✓		
Assertion [53]	App debugging, Flow rules inspection	✓		✓		
VeriFlow [66]	Verify and debug flow rules			✓		
Flover [54]	Flow policy verification, identify bugs in OF programs	✓	✓	✓		
OFTesting [55]	App testing and debugging	✓				
SE-Floodlight [56]	Role-based conflict resolution, authorization, security audit system		✓	✓	✓	
DDoSDetection [64], [84]	SOM-based DDoS attack detection		✓	✓		
HyperFlow [63]	Controller avialability		✓			✓
Min-Cut Placement [45]	Controller reliability, switch-controller connectivity		✓	✓		✓
Monitoring [83]	Data plane connectivity monitoring			✓		✓
FortNOX [65]	Find contradictions in flow rules, authorize applications		✓	✓		
FLowChecker [44]	Configuration analysis and verification		✓	✓		
DISCO [58], [59]	Controller availability, network monitoring		✓			
DCP [79]	Dynamic controller provisioning, scalability, availability		✓			✓
Ctrl-Placement [60]	Controller scalability and availability		✓			✓
Ctrl-Reliability [61], [62]	Controller reliability and availability		✓			✓
POCO [80]	Controller resilience and failure tolerance		✓			✓
Resonance [67]	Access control & dynamic policy enforcement			✓		
CPRecovery [42]	Controller resilience, switch connectivity, DDoS attack		✓	✓		

A. *Security Systems Development*

In SDN, network security is as good as the security policy. Security policies can be converged to security practices with the help of software applications or security modules in the control plane. Hence, security in SDN can be termed "*software-defined security*" and Scott *et al.* [5] termed SDN "*Security-Defined Networking*." Many of the security platforms mentioned in Table IV are OpenFlow applications, or SDN control platforms modified for security. In this section we discuss various mechanisms that are used to develop security systems for SDNs.

FRESCO [52] is a framework developed in-between the OpenFlow application layer and the OpenFlow controller that empowers developing security applications by accessing network flows, switch-wide statistics, and inserting new flow constraint rules to protect against detected threats. This framework simplifies the development of security applications by exporting an intermediate scripting API that enables secure access to network information. To strengthen a network defense system, FRESCO provides various built-in security modules for implementing essential security functions, such as firewalls, scan detectors, attack deflectors, IDS detection logic, and APIs to enable legacy applications to trigger these modules. Security applications developed in FRESCO can be deployed on any OpenFlow controller or switch implementations. It also enables OpenFlow and legacy security applications to co-work coherently for threats detection and mitigation.

Language-based security, showing how to program SDNs in a secure and reliable manner is presented in [91]. The authors propose a new programming model that supports network slicing in order to isolate traffic of one program from another or segregate one type of traffic from another within a program. Compared to traditional slicing approaches, this model provides principles for formal modular reasoning and enables programmers to maintain end-to-end desirable security properties within a slice. Similarly, the correctness of the compiler can be tested with a translation validation framework that automatically verifies the compiled programs that are developed for SDNs.

In [91], the authors discuss language-based security for creating secure network slices in SDN. A programming model is proposed that supports network slicing to isolate one type of traffic from another in order to maintain security and reliability in SDN. Similar to modules and abstract data types in programming languages, the proposed programming model enables network programmers to seal off portions of their SDN programs from outside interference. Therefore, a high-level abstraction with a well-defined semantics model that uses compilers to provide important security properties is proposed in [91].

A language for verifying and debugging SDN applications is presented in [53]. It is an assertion-based language that enables application developers to verify dynamic properties of controller applications using high level program statements. The Flow-based Security Language (FSL) [92] is a security policy implementation language that enables deployment of automatic Access Controll Lists (ACL), firewalls, and traffic isolation mechanisms. FSL extends the security notion beyond traffic blocking to traffic redirection, usage rate limitation, and security policy confederation.

A mechanism to define and detect interactions among various flow rules in an OpenFlow switch is presented in [93]. The authors proposed an algorithm for detecting interactions between OpenFlow switch rules and demonstrated its usefulness for developing OpenFlow applications.The tool demonstrated in [93] can be used to test flow rules-generating applications in order to avoid conflicting OpenFlow rules in the datapath.

B. *Flow Sampling*

In SDN, flow forwarding rules are installed in the flow tables of forwarding elements based on higher-level decisions in the controller. A flow of packets can be (re)directed to any destination (port) based on the flow rules in flow tables of a switch.

Flow sampling mechanisms comprise algorithms that select samples (e.g., packets or packet headers fields) of flows and send them to a desired destination. These samples can be used to check the contents of a flow, frequency of packets with particular characteristics, and inter-arrival times of various packets. In SDN, flow sampling is rather easy due to the centralized control on the packet forwarding behavior of various distributed forwarding elements.

To increase the visibility of packet contents at the controller, [94] propose per-flow sampling of active flows in the datapath. The controller defines the sampling rate at various granularity to retrieve samples of packets. In small networks where middle boxes would be costly, flow samples can be used by applications implemented in the controller, whereas in large networks the flow samples can be provided to middle boxes near switches to analyze various flows for security. The sampling parameters (e.g., number of samples per flow, etc.) can be set by the controller and performed in the data plane either stochastically (with a predetermined probability) or deterministically (based on some patterns).

The FleXam [95] framework proposes a sampling extension in OpenFlow switches to provide access to packet level information at the controller. FleXam enables the controller to define which packets or parts of packets (samples) should be sent to the controller, or security middle boxes or even forwarded to applications for security analysis, such as intrusion detection. In FleXam [95], a sampling action i.e., OFPAT_SAMPLING, similar to OFPAT_OUTPUT which sends the sampled packets to the controller, is assigned to each flow. These actions are based on six parameters, i.e., *scheme*, *p*, *m*, *k*, &, and *destination*. The scheme parameter defines the sampling scheme that is used to identify the parts of sampled packets that could be sent to a host. The m, k, and & parameters are used for deterministic sampling, and the p parameter is used for stochastic sampling. The destination parameter identifies the host where the selected samples will be sent, such as controllers, middle boxes, IDS, or other applications.

C. Content Inspection

Recent increase in volumes of traffic makes it difficult to inspect contents of each packet in contrast to high-speed requirements. SDN enables flow-level security measures that empower network security systems, since a flow of data is analyzed and selected packets are used for content inspection. Flow-based content inspection procedures enable cost effective deep packet inspection (DPI) in IDSs and IPS.

Intrusion detection and prevention systems (ID/PS) supervise the running status of networks according to security policies, finds attacks and threats, and deploy countermeasures in order to secure the network from future possible threats. In traditional networks, network nodes are independent and loosely coupled having no communication between their control planes. Hence, ID/PS in each network domain deal with their own challenges alone, and each node is configured independently. Similarly, each network branch has to be secured with its own ID/PS deployed at at each ingress and egress. These features make the current ID/PS systems complex, prone to errors, and costly.

SDN enables real-time deployment of novel ID/PS leveraging from the centralized control and visibility of the entire data plane. The SDN variant, i.e., OpenFlow, enables per-flow statistics collection enabling the controller to poll a switch for individual or aggregated flow statistics from the entire data plane. The controller can retrieve packet contents from the switch in order to provide packet contents to novel security systems implemented either as OpenFlow applications or security middle boxes. Similarly, traffic from a particular host or application can be re-directed to a security middle box at run time through changing the flow rules in a switch.

In [96], a fuzzy logic-based information security management system for SDN is proposed. The system [96] performs assessment of the security level of information, information security risk management, and intrusion detection and prevention. Implemented as OpenFlow application for the Beacon controller [97], the system consists of three modules, i.e., a statistic collection and processing module, and a decision-making module. Similarly, Learning Intrusion Detection System (L-IDS) using OpenFlow is presented in [98]. The authors claim that the proposed (L-IDS) [98] scheme provides rapid response and network reconfiguration in real time to secure embedded mobile devices.

An IPS based on SDN is presented in [99] which supports unified scheduling of security applications in a network and load balancing among the IPSs. In the proposed scheme, IPSs are registered with the OpenFlow controller that can check its availability, and redirect flows to various IPSs. SnortFlow [100] is a Snort-based [101] IDPS that enables cloud systems to detect intrusions and deploy countermeasures with the help of OpenFlow in run-time. SnortFlow combines the strengths of Snort [101], [102] for pattern matching, content analysis, and OpenFlow for network reconfiguration.

Frameworks which are based on flow sampling, such as [95] and [94], enable proactive security services deployment in SDN, and are simple enough to work at line rate. The four main features of these frameworks are easy switch implementation, low switch overhead, access to packet contents, and controllable network overhead. These frameworks enable anomaly detection with low network overhead and can be easily implemented in SDN due to the centralized control architecture in SDN. A Flow-based Intrusion Detection System (FIDS) in OpenFlow based on the NetServ programmable node architecture [103] is presented in [84]. Besides autonomic network management, the platform [84] demonstrates that FIDS can be used to protect a SIP application server against DoS attacks.

Deep Packet Inspection (DPI) techniques are used to examine application headers and payloads of packets in a network. DPI enables filtering network packets to examine their data part in order to find protocol non-compliance, viruses, intrusions, spams or other defined attributes. A cost-based virtualized Deep Packet Inspection mechanism is proposed in [104] that maintains an optimal tradeoff between minimum number of DPI engines and minimum network load. The proposed platform uses genetic algorithms to deploy DPI engines in order to optimize the cost of deployment, minimize its number and the global network load, and maximize the average number of flows for analysis.

D. Traffic Monitoring

Network traffic monitoring supports user application identification, anomaly detection, and network forensic analysis besides other fundamental network management tasks. Network traffic monitoring is a daunting task in traditional networks due to complexities in routing traffic to devices or tools for traffic analysis [88]. Since ISPs often do not have access to end-user devices, they prefer network layer measurements that use infrastructure elements, such as routers and switches to obtain statistics. Hence, the measurement granularity is often limited to port-based counters. This approach lacks the capability to differ between different applications and traffic flows, thus making network forensics difficult and complex [105].

SDN enables network traffic monitoring through programmability of the forwarding plane and novel control protocols between forwarding elements and the control plane. For example, the OpenFlow controller can retrieve switch statistics (e.g., OFPT_STATES_REQUEST and OFPT_STATES_REPLY), or extract sample packets from flows and provide them to monitoring applications for security analysis. The OpenFlow controller can also redirect network traffic to security middle boxes at runtime by updating the switch flow tables. Moreover, tools such as ndb [71] that performs packet back-trace, and OFRewind [72] that records and replays traffic, can be used to find the route of traffic origination to trace causes of traffic anomalies.

OpenSAFE (Open Security Auditing and Flow Examination) [88] is an OpenFlow-based network monitoring framework that enables network traffic monitoring and packet filtering at linerate. The OpenSAFE framework enables arbitrary direction of traffic to security systems and monitoring devices by introducing a language called ALARMS (A Language for Arbitrary Route Management for Security). OpenSAFE uses multiple parallel filters and route traffic to those filters with the help of ALARMs that creates flow paths and installs them on the switches.

The design of a multi-level security network switch system for OpenFlow is proposed in [106]. The system enables network traffic monitoring with the help of a packet filter that can check a packet's content, packet header fields and other packet attributes to restrict covert channels. Covert channels are designed in such a way that their very existence can be hidden, which can be used for malicious activities. Developed for OpenFlow networks, the system in [106] can implement multi-level security policies and restrict covert channels through traffic monitoring strategies.

PayLess, a network monitoring framework is presented in [107]. PayLess proposes a frequency adaptive statistics collection scheduling algorithm for network monitoring that provides real-time network information with minimal overhead. The flexible RESTful API of PayLess enables adaptive flow statistics collection for network monitoring at various aggregation levels. PayLess itself is a collection of pluggable components and can integrate custom-belt components with the help of well-defined interfaces.

The OpenNetMon [105] framework is proposed to monitor per-flow metrics, such as throughput, delay and packet loss, however, it can be used to provide flow statistics to security applications such as IDS and IPS etc. A traffic monitoring system for OpenFlow switches is demonstrated in [108] which can monitor all hosts and traffic that pass through the switches. The switch monitoring system provides warning functions to maintain traffic bounds on switch ports. The traffic monitoring mechanisms demonstrated in [108] can be used to detect flooding attacks and traffic anomalies in OpenFlow networks.

E. Access Control

Traditionally, access control mechanisms are enforced by firewalls which are deployed on network boundaries to examine all incoming and outgoing packets to defend a network against attacks and unauthorized access. However, conventional firewalls consider insiders to be trusted partners which is not a true assumption, since in-zone users could (for any reason) launch attacks or circumvent the security mechanisms. Moreover, dynamic changes in network policies, traffic conditions and complex configurations make the deployment of firewalls even harder.

SDN, however, makes the implementation of access control mechanisms rather easy to develop and deploy in the form of SDN applications. An OpenFlow firewall application implemented as a FloodLight module is available on the FloodLight community repository [109]. It enforces Access Control List (ACL) rules on OpenFlow-enabled switches by monitoring packet-in behavior. ACL rules comprise sets of conditions that allow or deny a traffic flow at its ingress switch. The firewall operates in a reactive manner, such that each new flow is compared against ACL rules and corresponding actions are taken.

FLOWGUARD [111] is an OpenFlow-based firewall framework that enables effective network-wide access control in SDN. FLOWGUARD [111] detects firewall policy violations and tracks the path of a flow to identify the source and destination of each flow in the network. This tracking capability helps in identification of flows having mutated packet header fields. A packet filtering application called ***oftables*** is presented in [111]. Besides providing basic firewall functions, *oftables* creates several perimeter networks to avoid insider attacks. A Firewall application that runs on the SDN controller as an application without the need of dedicated hardware is presented in [112]. The firewall proposed in [112] performs packet inspection and updates packet handling rules at run time. This firewall has a simple interface and a user can manage the flow rules from outside the controller.

F. Network Resilience

Network resilience strategies enable networks to maintain acceptable levels of network operation in the presence of challenges, such as malicious attacks, security breaches, misconfigurations, operational overload, or equipment failures [113]. Regarding network security resilience, security breaches must not limit the availability of network resources and information to legitimate users. Network resilience mechanisms are based on flexible configuration, and cooperation of interacting devices across the network that provide services such as flow monitoring, anomaly detection and traffic shaping, etc. A resilience strategy is described in [113] and defined as;

$D^2R^2 + DR$: Defend, Detect, Remediate, Recover, and Diagnose and Refine. This strategy puts forward the idea of a real-time control loop to enable dynamic adaptation of networks in response to faults and security challenges.

Two features of the SDN architecture can realize deployment of the above-mentioned control loops at run-time to enhance network resilience. First, SDN provides flexible architecture for quick and easy configuration of devices. Second, SDN provides control of the networked devices to a centralized controller that enhance cooperation between a variety of heterogeneous services and devices. The latter can also help in improving network resilience by orchestrating a variety of security services or applications deployed for specific services or network segments.

SDN-enabled network resilience management is presented in [114]. The framework is based on policy-controlled management patterns that can be described as a high-level description of the overall policy-based configuration and interactions between a set of resilience mechanisms. These mechanisms include mechanisms for detection of attacks and anomalies, and methods for the remediation of these challenges. The framework proposed in [114] is implemented as an OpenFlow application that translates high-level management patterns to low-level device configurations. Those configurations are translated to packet forwarding rules and using the OpenFlow protocol, those configurations are deployed in the OpenFlow switches as forwarding rules in its flow tables. One benefit of such architectures is that it can provide abstractions for orchestrating individual resilience services (OpenFlow applications).

The potential of using SDN/OpenFlow for disaster resilience in WAN is presented in [115]. Two important issues are evaluated, first, communication between networking devices and controllers, and second, recovery process after link failure. It is demonstrated that, if a link between a switch and a controller fails, link migration does not affect the throughput performance in the proposed framework. Besides that, the recovery process is very fast. Similarly, optimal disaster recovery in backbone networks with the help of SDN is presented [116]. The authors in [116] claim that SDN supports faster path recovery for multipath TCP (MPTCP) than legacy IP networks, since MPTCP can not reestablish the paths itself.

Similarly, using the centralized intelligence brought about by SDN, a mechanism for (software) component organization through SDN is proposed in [42] to enhance network resilience. The proposed framework [42] uses controller replicas to maintain network operation even if some parts of the network (including the controller) are compromised.

G. *Security Middle-Boxes*

Network traffic is routed to security middle-boxes in order to perform various network security functions. In traditional networks, middle-boxes have a number of challenges pertaining to middle-box placement, scalability and security policy alteration. Joseph *et al.* [117] attribute these challenges to complex manual configurations, the need of on-path deployment of middle-boxes in existing networks, and inflexibility in current networks. SDN makes the deployment of middle-boxes simple and elegant through network programmability and centralized network control. In [118], it is presented that there is no need of deploying function-specific middle-boxes in SDN, rather, integrate its processing into the network itself. Authors of [119] describe how SDN simplifies the management of complex and diverse middle-boxes in software-defined middle-box networking.

Anwer *et al.* proposed a control plane, *Slick*, for network middle-boxes in [120]. The decoupling concepts of SDN are used to separate the processing between in-network middle-boxes and a controller that coordinates those devices. The Slick control plane for middle-boxes supports heterogeneous devices, their dynamic reconfigurations, and automatic traffic steering and function placement with migration capabilities. Similar to OpenFlow, Slick provides a control plane protocol to install or remove code from middle-boxes, a Slick controller that determines where to install elements (middle-boxes), and a programming model that simplifies policy implementation as applications and encourage code reuse.

The SIMPLE [89] framework provides an SDN-based policy enforcement layer to manage middle-box deployments and middle-box-specific traffic steering. The policy layer translates high-level middle-box policies into efficient and load-balanced data plane configurations that steer traffic through a sequence of middle-boxes. One of the main contributions of this framework is that it requires no modification neither in the SDN architecture (e.g., OpenFlow protocol and OpenFlow switches), nor in middle-box functionalities. The FlowTags [121] architecture uses packet tags that hold the middle-box context. The SDN controller configures switches and middle boxes to use these tags in order to enforce network-wide policies.

Optimal Security Traversal with Middle-box Addition (OSTMA) in OpenFlow networks is presented in [122]. Security traversal services route data flows to security devices or middle-boxes which can introduce delay. In [122] the security traversal path is modeled as a constrained shortest path problem (CSP) to provide deterministic delay guarantee and maintain minimum transmission costs. A middle-box monitoring module is developed for the OpenFlow controller and a security traversal engine is implemented in the OpenFlow controller. Based on the network condition information in the controller, the best security traversal path (minimum cost path among the alternatives) is configured in the data plane. Hence, [122] provides mechanisms for efficient use of middle-boxes in terms of security and cost penalty.

These proposals and approaches can be used to introduce security middle-boxes in SDN. However, the centralized control, programmability of network elements, and openness to application development in SDN might curtail the need of security-specific middle-boxes (at least in small and medium enterprise networks). The sampling approaches described in the previous section enable applications to use samples of packets at any time and perform security analysis. These applications can stop malicious traffic by installing or modifying forwarding rules in switches through the SDN controller. Furthermore, the placement and integration of security middle-boxes in SDN needs thorough analysis for two reasons. First, it might increase delay in network operation due to contacting the controller for suspicious traffic. Second, the increased number of queries in the controller might have a controller performance penalty.

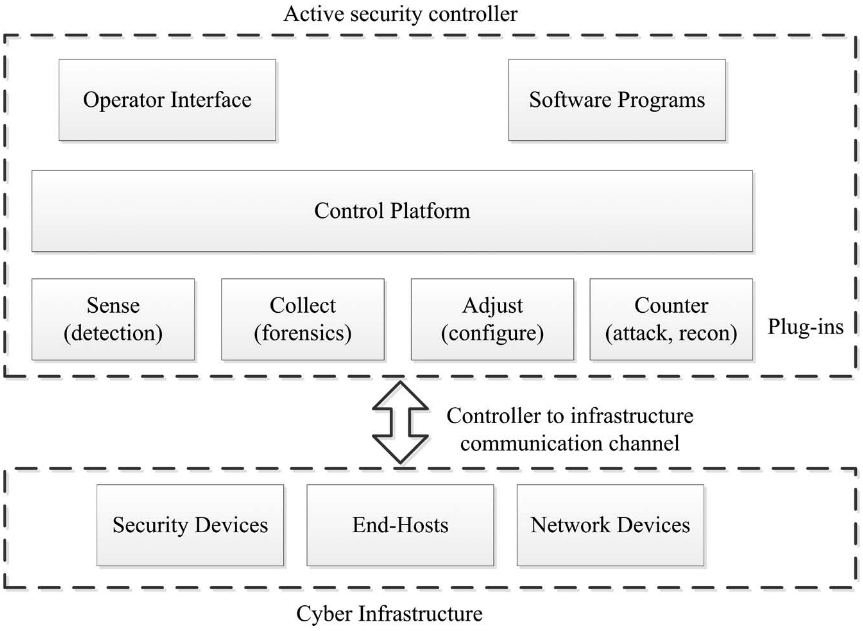

Fig. 4. The active security platform.

H. Security-Defined Networking

Besides, function-specific security systems in SDN such as firewalls, security middle-boxes and intrusion detection/prevention systems, there are unified security architectures that combine the functions of each of the above security systems for network-wide security. Using the centralized control plane and programmable network devices, various network security architectures are proposed to provide robust network-wide security. Below, we describe various SDN-based networking architectures in which network security is an inherent feature of the architecture.

Active security [123] provides a centralized programming interface to control detection of attacks, data collection for analysis of attacks, configuration revision, and reaction to attacks. It couples passive components for monitoring the state of the network with highly dynamic components for enforcing policy or manipulating traffic, and a programming environment for exercising granular control over each of these. The Floodlight [73] OpenFlow controller is modified for Active security [123] so that it can interact with end-host systems and security middle-boxes, and work as an active security controller. The controller has interfaces to network equipment, end-hosts, and security systems allowing for programmatic control over an event-driven feedback loop of the entire cycle of configuration, detection, investigation, and response. The active security platform architecture is shown in Fig. 4.

The core capabilities of active security are based upon five security measures and components. First, protection through configuration of the infrastructure using protection mechanisms which provide security against common attack scenarios to build a context-aware security system at the next level. Second, detection through interfaces to different sensors or sources that perform some detection and monitoring such as IDS to notify the reactive security system. Third, adjusting the network dynamically by using the SDN controller to better defend or monitor the network in future. Fourth, collection of attack statistics to perform forensics evidence gathering (e.g., memory gathering) in order to understand the attack and attribute it to an individual or organization. Fifth, reconnaissance and counter-attack by closely monitoring the attack using mechanisms such as honeypots, and responding by launching a counter-attack such as denial of service to consume the attacker's resources and limit his ability to continue attacks.

OrchSec [124] is an orchestrator-based architecture for enhancing network security using network monitoring and SDN control functions. The authors of OrchSec [124] have proposed decoupling of the control and monitoring functionalities in SDN. In OrchSec, the controllers are responsible only for issuing control messages (e.g., flow rules), whereas network monitors are responsible for doing monitoring functions. Similarly, applications are not implemented inside the controllers but rather developed as Northbound applications to provide independence to applications from the underlying controller architectures. OrchSec provides monitoring functionalities at different granularity (e.g., varying sampling rates, changing responsibilities between controllers and monitors) based on network-wide security requirements.

LiveSec [125] is an OpenFlow-based agile architecture for network security management in production networks. The LiveSec framework can visualize a network environment to enable event replay besides interactive policy enforcement and application-aware monitoring. The LiveSec architecture comprises a LiveSec controller, Access Switching switches (OpenFlow switches), and legacy switches. The legacy switches perform layer-2 switching and are connected to the Access-Switching layer as shown in Fig. 5. Various security service elements can be attached to OVSs, for example in the VM service elements. When a user attempts to access the network from outside through the Internet gateway, the traffic can be directed to security service elements after the first flow enters the

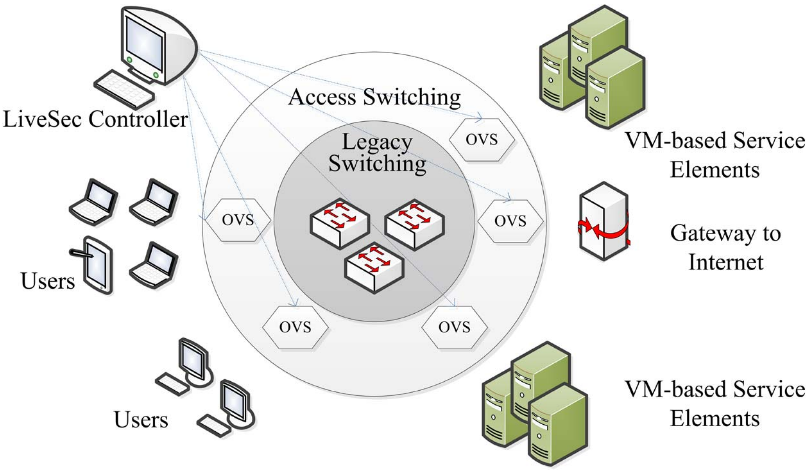

Fig. 5. The LiveSec architecture.

TABLE V
IMPLEMENTATION AND DEPLOYMENT COSTS OF SECURITY SOLUTIONS

Security solution	Security Type	Implementation Framework	Costs						
			Additional Components			Modifications			
			Soft.	Ctrl.	Sec-Dev.	Ctrl	Data	OF Prot.	Hosts
FleXam [95]	Flow sampling	OF extension	No	No	No	Yes	Yes	Yes	No
Sampling [94]	Per-Flow sampling	OF extension	No	No	No	Yes	Yes	Yes	No
SnortFlow [100]	IPS	Snort [101]	Yes	No	Yes	Yes	No	No	No
FIDS [84]	Flow-based IDS	NetServ [103]	No	Yes	Yes	Yes	Yes	Yes	No
FLOWGUARD [110]	Firewall	Access control	Yes	No	No	No	No	Yes	No
SE-Floodlight [109]	Firewall	Access control lists	Yes	No	No	No	Yes	No	No
OFTABLES [111]	Firewall	Flow rules filter	Yes	No	No	Yes	No	No	No
SoftFirewall [112]	Firewall	Flow rules filter	Yes	No	No	No	No	No	No
MangementFramework [114]	Network resilience	Management patterns	Yes	No	No	No	Yes	Yes	No
CPRecovery [42]	Network resilience	Backup mechanisms	Yes	Yes	No	No	Yes	Yes	No
OrchSec [124]	Security monitoring	Security orchestrator	Yes	Yes	No*	No	No	No	No
Multi-Level Sec [106]	Network monitoring	Covert channel filter	No	No	Yes	Yes	No	No	Yes
LiveSec [125]	Network monitoring	Sec. management system	Yes	Yes	No*	Yes	No	No	No
PayLess [107]	Network monitoring	Monitoring framework	Yes	No	No	Yes	No	No	No
OpenNetMon [105]	Networking monitoring	Monitoring framework	Yes	No	No	Yes	No	Yes	No
FortNOX [65]	Flow rules verification	Role-based authorization	Yes	No	No	Yes	No	No	No
FLowChecker [44]	Configuration verification	ConfigChecker tool	Yes	No	No	Yes	No	No	No
DISCO [58], [59]	Controller availability	Distributed ctrl-plane	Yes	No	No	Yes	No	No	No

*Does not require a special device, however, the architecture can efficiently use security devices

network. Since, every first packet of flow in SDN is directed to the controller, user authentication and security service activation can be performed automatically.

I. Costs Analysis of Security Platforms

Network security systems consume network bandwidth, processing capacity and memory, together with performance penalty in terms of complexity, and latency. In this section, we provide a brief overview of the costs associated with security solutions used in SDN. For brevity, the costs are divided into two broad categories. First, costs of developing and deploying a new security system that would require expenses on solution-specific security applications, security controllers and/or security solution-specific devices. Secondly, costs involved in adding security modules in OpenFlow controllers, data path elements, and modifications to the OpenFlow protocol and hosts or end user devices. Table V presents various solutions with associated costs, and indicates whether a security solution requires new security elements or is based on modifications to existing SDN elements.

In SDN, one can argue that network security costs would be higher if a security system requires solution-specific devices, such as security controllers, and solution-specific security devices. Taking benefit from the centralized control and network programmability, it is understandable that adding a security module to an OpenFlow controller or a data path element would be comparatively less costly. On the contrary, merging security solutions in the controller will raise challenges of controller availability and scalability. Therefore, the costs of security solutions depend upon the size and architecture of a network, as well as, bearable performance penalty.

Besides the direct costs of implementing and deploying a security solution, it is necessary to evaluate performance penalty costs, such as complexity of deployment and use, latency, and network overhead. For example, consider passive and active measurement schemes for network monitoring. Passive measurement methods measure network traffic by observations while active measurement methods inject additional packets into the network to monitor their behavior. Thus, passive measurements require synchronization between observation beacons that complicate the monitoring process, whereas active measurements induce additional traffic load that affects the network and influences the accuracy of the measurements themselves [105]. In doing so, passive measurements increase delay, whereas active measurements increase the overhead by adding extra packets.

To illustrate the performance penalty, consider the inspection time of the SE-Floodlight [109] controller and FlowGuard [111] built on top of the Floodlight controller. The SE-Floodlight controller has a security module operating as a firewall, whereas FlowGuard is an application built on top of the Floodlight controller. The inspection time of 90% packets is approximately 40 μs in SE-Floodlight, whereas the inspection time for the same percentage of packets in FlowGuard is 79 μs. In a nutshell, this means that an isolated application consumes more time on packet inspection than a built-in module in the controller. Thus, there is always a performance trade-off that must be evaluated for each network setup.

Compared to traditional networks, SDN minimizes the costs of network security from many perspectives. First of all, SDN enables software-based solutions that diminishes the necessity of security-specific devices. Therefore, the Capital Expenditure (CAPEX) could be minimized. Moreover, SDN brings forward programmability in networking which eliminates the need of per-box manual configurations and that minimizes Operational Expenditures(OPEX). Therefore, in a broader view when compared to traditional networks, SDN minimizes both CAPEX and OPEX on network security.

VII. SDN-Based Virtual and Cloud Networks Security

Virtualization is used to decouple a system's service model from its physical realization and has been used in networking e.g., for creating virtual links (tunnels) and broadcast domains (VLANs). Through virtualization, logical instances of physical hardware can be used for different tasks where the physical and logical instances are mapped through a network hypervisor [126]. Cloud computing uses remote servers over the Internet to maintain data, services, and applications by allowing operators and subscribers to use them without maintaining their own infrastructure for computing and storage. Cloud networks bring technologically distinct systems onto a single virtualized domain on top of which services could be deployed to achieve a high degree of service availability and flexibility. The importance of virtual and cloud networks leveraging SDN for future mobile networks with its challenges and benefits is presented in [127], [128].

However, there are daunting security challenges in virtual and cloud networks. These challenges are explained below with the state of the art in SDN-based security solutions.

A. Virtual Networks Security

Virtualization enables multiple tenants or network users to share the same physical network resources that can create security vulnerabilities. A literature study on security implications of virtualization [129] shows that virtualization has a positive effect on availability but has threatening security challenges related to confidentiality, integrity, authenticity and non-repudiation. Virtual machines can be created, deleted and moved around a network easily, hence, tracking a malicious virtual machine would be much more complex. Similarly, if a hypervisor is hijacked the whole system can be compromised [130].

SDN provides a layered design that hides the complex hardware from SDN applications implemented on top of the control plane. SDN offers standard interfaces between control plane applications and forwarding elements, and is thus considered to be a natural platform for network virtualization. Network hypervisor, a program that provides an abstraction layer for the network hardware, enables network engineers to create virtual networks that are completely decoupled from the network hardware. FlowVisor [131] is one example of hypervisors that uses OpenFlow and sits between software controllers and hardware switches to filter events to the controller and mask messages to switches.

FlowVisor [131] enables network administrators to identify and differentiate network slices with the help of packet header fields. From the security perspective, FlowVisor can be used to isolate different types of traffic and provide an interference-free environment to maintain privacy and confidentiality in communication networks. Isolation is enforced even under adversarial conditions so that one slice is not able to exhaust the resources of another [131]. For example, network hardware has low-power embedded processors and finite memory that can be prone to resource exhaustion or saturation attacks. FlowVisor tracks the processing plane and explicitly lists flow entries in a switch so that a slice does not exceed preset limits.

A Network Intrusion detection and Countermeasure sElection (NICE) framework for virtual network systems is proposed in [132]. The NICE framework is a multiphase distributed vulnerability detection, measurement, and countermeasure selection system that can be used to detect and mitigate collaborative attacks in a cloud virtual networking environment. The NICE platform is built on graph-based analytical models and reconfigurable virtual network-based countermeasures to prevent vulnerable virtual machines from being compromised in cloud networks. The NICE framework leverages OpenFlow network programming APIs to build the monitoring and control plane system over distributed programmable virtual switches.

B. Cloud Networks Security

Cloud computing systems comprise various resources which are shared among users with the help of hypervisors. From a security point of view, it is possible that a user spread malicious traffic to tear down the performance of the whole system, consume more resources or stealthily access the resources of other users. Similarly, in multi-tenant cloud networks in which tenants run their own control logic, interactions can cause conflicts in network configurations. However, given the unified view of all the resources to a centralized control plane system of SDN, these challenges can be efficiently solved. The rise of SDN will change the dynamics around securing the data centers by offering opportunities to research for enhanced security [133].

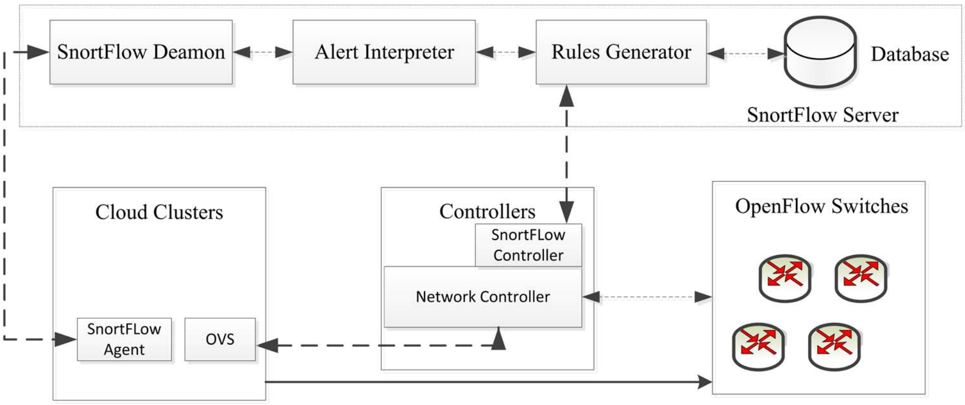

Fig. 6. SnortFlow system architecture.

Presenting SDN security considerations in data centers, a use case of Automated Malware Quarantine (AMQ) in SDN is presented in [133]. AMQ detects potential network threats and isolates unsecure network devices to stop them from negatively affecting the network. AMQ uses software patches to resolve threats and automatically allows quarantined devices to join the network. However, today's AMQ solutions consist of proprietary devices which have limited awareness of other devices and are designed for static traffic flows. Therefore, AMQ presented in [133] is implemented as an OpenFlow application that leverages from the centralized control plane for network awareness and adjustment to dynamic traffic flows.

The AMQ implementation using SDN [133] consists of two primary security Network Services Modules (NSM), i.e., the Bot Hunter NSM and the Threat Responder NSM. The Bot Hunter NSM monitors the network in real time and detects Malware-infected hosts. The Threat Responder NSM initiates the quarantine procedure with the help of the controller to isolate the threat from the network. This AMQ system can apply security policies to individual switch ports transparently and dynamically. The automatic configuration enabled by SDN reduces the response time to security threats, puts an end to manual configurations to deploy a policy, eliminates the need of hardware in small and medium enterprise networks, and minimizes CAPEX and OPEX [133].

CloudWatcher [134] is an SDN monitoring application implemented on top of the control plane to provide monitoring services to large and dynamic clouds. CloudWatcher controls network flows to guarantee their inspection through security devices and provides a simple policy scripting language to use the services. CloudWatcher leverages SDN to dynamically control network flows and change routing paths of flows so that intended flows pass through security devices. It consists of three main components, i.e., the device and policy manager that manages the information of security devices, the routing rule generator, and the flow rule enforcer to enforce the generated flow rules in switches.

SnortFlow [100] is a Snort-based IPS which enables the cloud system to detect intrusions and deploy countermeasures by reconfiguring the cloud networking system at run-time. Besides OpenFlow controller and switches, SnortFlow consists of Cloud Cluster that hosts cloud resources and the SnortFlow agent, and SnortFLow Server that evaluates the network security status and generates actions for the controller. The SnortFlow server has a SnortFlow daemon that collects alert data from the Snort agent, an alert interpreter that parses and targets suspected traffic, and a rules generator that generates rules for reconfiguring the network. These rules are stored in a database for future operations as shown in the system architecture in Fig. 6.

VIII. SDN Security According to ITU-T Recommendations

To address all the aspects of network security, network security dimensions are proposed by ITU-T in its security recommendation [26]. The security dimensions consist of a set of security measures to protect against all major security threats. In this section, SDN security platforms and mechanisms are discussed according to the ITU-T proposed security dimensions presented in Fig. 7. The security solutions are arranged in Table VI against each of the security dimension. Critical remarks in Table VI against each dimension present the lack of stable security solutions for that dimension. Similarly, "*Open Challenge*" shows that there are no compelling solutions addressing that security type directly. SDN security solutions are described below for each of the security dimensions.

A. *Access Control*

The access control security measures ensure that only authorized personnel or devices access the network resources. Unauthorized access to the controller or an application server that stores user credentials can spread havoc across the network. Unauthorized access to OpenFlow switches can lead to security issues, such as cloning or deviating traffic (for theft purposes), sending forged flow requests to controllers (DoS attack), and injecting malicious flow rules in switches. PermOF [34] secures the network resources from malicious applications by implementing a customized permission system and run-time isolation of OpenFlow applications.

Flow Security Language (FSL) [92] enables dynamic access control policy deployment in SDN using network flows. FSL improves deployment of multiple policies and mitigates the risks of policy conflicts. Resonance [67] is a system developed to enforce dynamic access control policies based on real-time alerts and flow-level information. Resonance provides mechanisms to directly implement dynamic network security

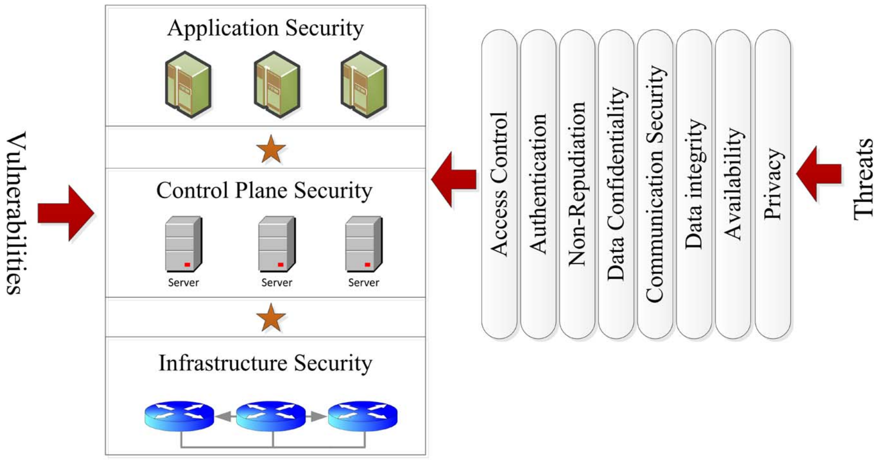

Fig. 7. Applying ITU-T security dimensions to SDNs.

TABLE VI
SDN SECURITY SOLUTIONS ACCORDING TO ITU-T SECURITY RECOMMENDATIONS

Security Type	Solution Name	Mechanism used	Remarks
Access Control	PermOF [34]	Impose access control on OF apps	*Access control also needed for application plane and multiple controllers.*
	FRESCO [52]	Enables develop security architectures for ACL	
	FSL [92]	Access control policy enforcement framework	
	Resonance [67]	Enables dynamic access control policies	
Authentication	FortNOX [65]	Role-based authentication & authorization	*Open Challenge.*
	FSL [92]	Controls authentication policies (admission control)	
Non-Repudiation	[135]	Uses permanent user identities (LISP)	*Open Challenge.*
	OFHIP proposed in [136]	Uses HIP for permanent identities	
	VAVE [137]	Source address validation of incoming packets	
Data Confidentiality	OF-RHM [138]	Random host mutation	*No specific security systems or applications.*
	FortNOX [65]	Data confidentiality through flow rules-legitimacy	
	IBC [139]	Identity-based cryptography	
Communication Security	TLS [46]	Ensure controller-switch communication security	*Complex Configurations.*
Data Integrity	[91]	Traffic isolation-based integrity	*Identity management systems are lacking.*
	OFHIP [136]	IPSec encapsulated security payload (ESP)	
	Other	VeriFlow [66],FortNOX [65], etc. ensure integrity	
		through flow rule legitimacy.	
Availability	DISCO [58], [59]	Distribute SDN control plane	*Less research efforts to increase availability through higher security. App. plane & data plane availability is still a challenge.*
	McNettle [74]	Extended processing capabilities	
	[12], [74]	Parallelism in multi-core processors	
	Other	Control-data plane functionality trade-off	
		and optimal controller placement strategies	
Privacy	OF-RHM [139]	OpenFlow random host mutation	*Systematic user privacy enforcement mechanisms are lacking.*
	Isolation [91]	Traffic-isolation-based privacy	
	ident++ [140]	User-selected security procedures	

policies on the device level while leaving little responsibilities to higher layers. It also implements network-layer security policies and provides a monitoring control interface which is required to control traffic according to predefined policies.

B. Authentication

Authentication security mechanisms ensure the identities of the communicating parties and that a user or device is not attempting a masquerade or unauthorized replay of previous communications. In SDN, the application server needs to authenticate devices and users before providing information such as user identities or credentials. Applications must also be authenticated before access is provided to the controller interface or network resources. The data plane also needs mechanisms to authenticate the controllers to avoid false rule insertions. If multiple controllers are used, the switches should be capable of authenticating them and maintaining the necessary controller redundancy. The latest OpenFlow switch specification i.e., OpenFlow v.1.3 specifies mechanisms for access to multiple controllers.

FortNOX [65] is a software extension to NOX that provides role-based authorization and security constraint enforcement for the NOX OpenFlow controller. This approach prevents an adversary attempting to strategically insert flow rules that would otherwise circumvent flow constraints imposed by the OpenFlow security applications. The role-based source authentication in FortNOX recognizes three authorization roles among which right to add, modify, and remove flow rules are distributed. The human administrator role gets the highest priority to insert flow rules, and attributes of the highest priority flow rules are sent to the switch. Similarly, security applications and then non-security-related applications get the priorities. Moreover, different authentication schemes [141] can be chosen according to the network architecture and system capabilities, and deployed and relegated through the centralized control plane.

C. Non-Repudiation

To ensure that a particular action has been performed by a specific user or device is non-repudiation. Proper identities are used to ensure that an authentic user or device can access particular services and resources. The controller must keep track of identities of applications making changes to or accessing the network resources for different functionalities. The controller also needs to associate proper identities with the forwarding devices to mitigate the risks of false and malicious requests. The switches need to keep proper identities for the legitimacy of the controllers. Encryption and identity management mechanisms can be used to prevent non-repudiation vulnerabilities.

In [135], the use of the locator/identifier separation protocol (LISP) [142] is proposed for maintaining accountability in OpenFlow networks. The permanent identifier in LISP, which does not change with changes in location, is used for user identification. Similarly, OpenFlow-Host Identity Protocol or OFHIP proposed in [136] introduces a cryptographic namespace to ensure user identity. Source address validation of all incoming packets presented in VAVE [137] can ensure the identities of sources and users and prevent security lapses.

D. Data Confidentiality

Data confidentiality security mechanisms protect the data from unauthorized access. Encryption, access control mechanisms and file permissions are used to ensure data confidentiality. In order to prevent attacks by impersonating a switch or a controller, OpenFlow has an optional security feature to use TLS where identification certificates are properly checked in either direction and allows encrypting the control channel in order to prevent it from eavesdropping. In [138], a moving target defense (MTD) technique called OpenFlow Random Host Mutation (OF-RHM) is proposed which mutates IP addresses of end-hosts to avoid scanning attacks. The mechanism for enforcing slice isolation-based confidentiality is demonstrated in [91]. Focusing on the packet processing functionality, the traffic of multiple slices is isolated through algorithms specifically designed for compiling slices to avoid interference.

FRESCO [52] and FortNOX [65] provides mechanisms to ensure data confidentiality by flow rules legitimacy. The use of Identity Based Cryptography (IBC) considering the controller as the trusted third party for secret generation is demonstrated in [139]. The proposed mechanism ensures data confidentiality in a hybrid SDN environment. Host Identity Protocol (HIP) [143] provides cryptographic identities and enhance data confidentiality through authentication mechanisms involving consistent host identifiers (HI) or host identity tags (HIT). The possible use of HIP in OpenFlow architecture for increasing data confidentiality besides mobility is demonstrated in OFHIP [136].

E. Communication Security

Communication security is necessary to ensure that the data flows between the authorized end-points and is not diverted or intercepted in between. OpenFlow defines Transport Layer Security (TLS) [46] and Datagram Transport Layer Security (DTLS) [47] to secure communication between controllers and switches. The TLS protocol provides privacy and data integrity between two communicating parties. TLS is composed of two layers i.e., the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides connection security ensuring connection privacy and connection reliability. The TLS Handshake Protocol enables a server and a client to authenticate each other, and negotiate encryption algorithms and cryptographic keys [46]. The DTLS protocol is used to secure data between communicating applications. It runs in the application space and is designed for UDP traffic [47].

Multiple channels (associations) between switches and controllers are suggested since a single channel might result in service outages due to connection failures. The latest OpenFlow specifications support multiple in-band or out-of-band connections between switches and controllers. Hence, this option should be used to improve network resilience in case of link failures. A fast link restoration mechanism has been proposed and demonstrated in [144]. The authors suggest backup entries with different priorities in the OpenFlow switches. These backup paths are computed by the controller after computing the working paths. Hence, upon link failure, the traffic is locally switched on a backup path without involving the controller. Similarly, flow entry migration techniques such as the ones proposed in [145] can reinstate a flow within 36ms that fulfills the carrier grade recovery requirement of 50ms. Moreover, HIP [146] based scheme proposed for SDN-based mobile networks in [147] can secure the control channel between the control and data planes.

F. Data Integrity

Data integrity security ensures the correctness or accuracy of data in transmission and protects it from unauthorized modification, deletion, creation and replication. Naturally, SDN ensures data integrity through verifiable flow rules, data origination and destination visibility, virtualization techniques, and per-flow security analysis. FortNOX [65] provides role-based authentication for determining the security authorization of OpenFlow applications. It extends NOX to support digital signatures and use an alias-set rule reduction algorithm to avoid flow-rule contradictions to ensure that the data is forwarded to the legitimate user or device.

VeriFlow [66] puts forward a proactive mechanisms to inspect the flow rules dynamically and in run time to maintain the integrity of the flow rules. OFHIP [136] employes IPSec encapsulated security payload (ESP) in transport mode for protection against DoS attacks, data origin authenticity, connectionless integrity and anti-replay protection. Traffic isolation-based integrity is proposed in [91]. However, no specific security platforms are devised to ensure data integrity, or that has data integrity as one of its main security goal.

G. Availability

Availability ensures that there is no denial of authorized access to network resources and applications. Events impacting the network, such as system failures or disasters, scalability and security compromise, must not limit access to authorized users and devices. Hence, availability has multiple dimensions, and due to the centralized control plane, there are many research proposals to increase the scalability and security of the SDN controller. The platforms for controller scalability are discussed in Section V-B. There are some proposals that increase controller availability by minimizing controllers' responsibility such as DevoFLow [148] and DIFANE [149]. Devolved OpenFlow or DevoFLow [148] enable scalable flow management by aggressive use of wild-carded OpenFlow rules. DIFANE [149] proposes to relegate the controller to simpler tasks of generating rules but not involved in real-time handling of data-packets.

High availability also needs fast restoration from failures. A fast restoration mechanism for an OpenFlow network is discussed in [150] where a flow can be migrated to another path within a very short interval on the directions from the controller. Similarly, HotSwap [151] is a system for upgrading or replacing a controller without service disruption and minimal overhead. Centralized architectures also increase the administrator's responsibilities where the administrator can become a bottleneck due to lack of knowledge or availability etc. Mechanisms for devolution of the administrator's responsibilities is proposed in ident++ [140], where the end-users and applications can set their traffic rules and security mechanisms. The source address validation of all incoming packets presented in VAVE [137] for the OpenFlow architecture is another approach to stop unwanted traffic and mitigate the risks of controller and switch resource exhaustion.

In OpenFlow, it is necessary to ensure the availability of the switch flow tables for new incoming requests. Sizes of flow tables might be limited which can cause the switch to drop legitimate flow requests. Another challenge that requires attention is ensuring availability of the application plane. Services offered by operators might reside in a cloud network which must be available to users. This challenge will be more visible when commercial networks are converged into SDNs.

H. Privacy

Privacy mechanisms ensure protection of information which might be derived from observing the network activities. Transport Layer Security (TLS) is specified for the OpenFlow communication which is a common approach for enabling secure communication in client/server applications on the Internet. TLS is mostly used to prevent eavesdropping, tempering or message forgery. Generally, its primary goal is to enable privacy and data integrity between the communication parties through the use of symmetric cryptography for data encryption. The protocol is composed of two layers; TLS Record Protocol and TLS Handshake Protocol. The Record Protocol guarantees connection privacy and reliability by means of data encryption supported by TLS handshake protocol, and MAC to ensure integrity. The TLS Handshake part is responsible for authenticating the communicating parties with each other, and to negotiate the encryption algorithm and cryptographic keys before transmitting the first packet of an application.

Besides the use of TLS and DTLS as specified in the OpenFlow specification, virtual networks or network slicing can be used to provide isolation-based information privacy. Individual slices are separated by a networking hypervisor such as the FlowVisor [131]. The Open vSwitch platform provides isolation in multi-tenant environments and during mobility across multiple subnets [152]. The VAVE [137] platform enables information privacy and prevents data from being spoofed or forged through the OpenFlow interface attached to legacy devices.

The ident++ [140] approach can increase user privacy by allowing users to enforce their own security procedures. Similarly, the OF-RHM mechanism [138] increases user privacy by changing end-hosts' IP addresses randomly and frequently. A framework called Enterprise-Centric Offloading System (ECOS) [153] leverages SDN to provide secure offloading of mobile applications in enterprise networks. The ECOS framework offers varying levels of trust and privacy to corporate mobile users to access their systems.

IX. Future Directions

SDN enhances network security due to global visibility of the network state, centralized intelligence and network programmability. As such, a common distribution layer gathering information about security requirements of different services, resources and hosts, and disseminating the security establishing commands to the network elements to enforce security policies can result in robust and scalable security enforcement. However, the same key attributes of SDN i.e., centralized intelligence and programmable network elements, make security in SDN a more challenging task. Hence, many potential SDN security solutions and platforms are proposed that are described in previous sections.

To our understanding, there are still some gray areas which need to be understood and properly addressed before SDNs are commercially deployed. Henceforth, we point out some research challenges and future directions below.

A. Programming and Development Models

SDN enables developers to develop novel networking architectures, protocols, applications, and test or use them in operational networks. This capability will bring innovation in networking but can also introduce security challenges with the myriad of possible new applications running in a network. The independence and loose interconnection of various development environments and arbitrary control platforms that will deploy the functionalities of independent applications on possibly the same forwarding elements can create severe security challenges. This threat can be visualized with the help of Fig. 8.

As can be seen in Fig. 8, if an application is developed in an environment that uses the control plane that is developed in a different environment, the resulting functionality taken from the same data plane might induce security vulnerabilities besides other challenges, such as security policy collision and difficulty in integration. Implementation is a key factor that induces security vulnerabilities in Internet applications [154]. Therefore, proper programming models and paradigms, and development

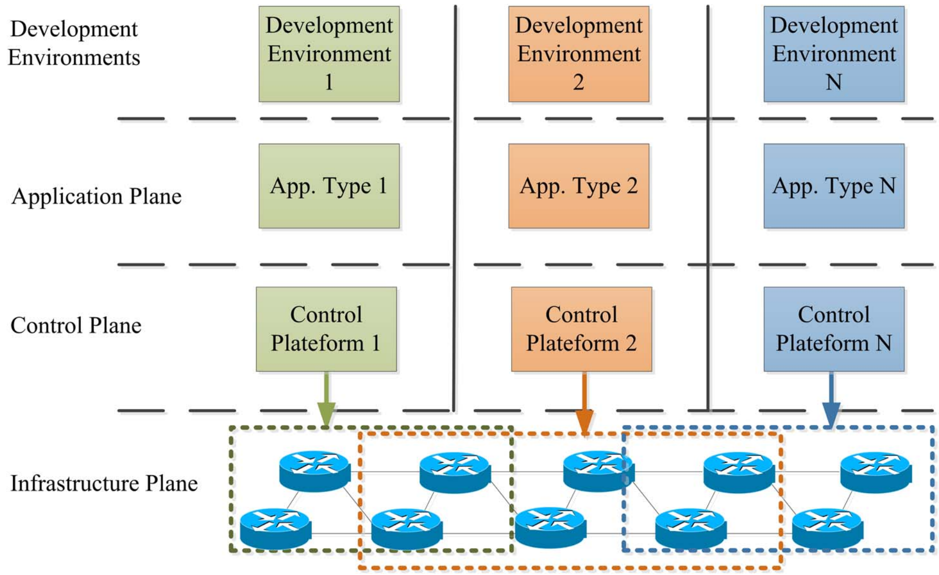

Fig. 8. Current trends of development in SDN.

environments must be standardized to minimize the chances of conflicting modules that create security vulnerabilities. Conflicting modules in distributed systems can create security vulnerabilities, such as exposing sensitive network information or APIs, and creating contradictory flow rules.

B. Class-Based Application Security

In SDN, most of the networking functionalities will be implemented in software as applications on top of the control plane. Consider that some applications might need network statistics for load balancing, some applications might need samples of packets, and so on. Each of these applications has its own distinct functional requirements and each of these applications should be provided differentiated access. Similarly, some applications might need their traffic to follow a path with particular characteristics, e.g., avoid some links, stay within a certain jurisdiction and use a path within the cost constraints, as described in [35].

In SDN, security applications seek network statistics, samples of packets and access network resources through the control plane. A monitoring application that works on fine-grained tracking of packet behaviors in the data plane require direct view of the network. However, the control plane provides networks statistics to applications based on its own view which might be incoherent with the actual network view. Hence, some applications need direct access to network statistics and resources. On the other hand, in the current trends of apps. development, providing direct access to a variety of applications might create security challenges.

Therefore, it is necessary to categorize applications into classes or groups and enforce security procedures based on high-level security policies defined for each of the class. This will require first to *Categorize* applications into *Classes* according to their functions and their requirements from the underlying network resources. Secondly, security policies need to be defined for each class that will invoke security procedures upon an application's request for access to network resources (e.g., flow tables in OpenFlow switches) or network information (e.g., network statistics). Having defined such classes of applications, specific security procedures based on high-level security policies can be enforced on an aggregated sum of applications.

Three classes of applications significantly impacting the network security in SDNs discussed in the IETF draft [35] could be listed as "*Network Sensitive Applications*," "*Services for the Networks*" and "*Packaged Network Services*." Authenticating and authorizing each application in the same way will make the control plane a bottleneck due to large numbers of incoming requests for access to network resources or with attempts to derive network statistics. The categorization of applications would enable lenient per application-class based security enforcement, and involve minimal controller overhead to authenticate and authorize applications based on security policies compiled for various classes of applications.

C. Scalability and Security

Scalability is one of the major challenges faced by the logically centralized SDN architecture. In SDNs, as the network size and diameter grows, the amount of control traffic destined towards the centralized controller increases and as a result the flow setup time grows [63]. Moreover, it is known that the capability and operation set of an OpenFlow controller is most likely limited. Hence, the lack of scalability in SDN can enable targeted attacks by inundating communication between the controller and the switch to cause control plane saturation [39]. Besides saturation of the control plane, it is demonstrated in [41] that a network can be compromised by exhausting resources of OpenFlow-enabled switches.

Availability is a security dimension that strongly correlates scalability with security. Most of the security threats in SDN target to compromise availability of the control plane. Thus, multiple controllers are suggested, but, simply adding multiple controllers might result in cascading failures of controllers as demonstrated in [40]. Therefore, it it is necessary to correlate security and scalability in SDN to design secure SDN architectures that ensure high availability of the control plane. For the data plane, load balancing and connection migration techniques could be used to equally share the load among OpenFlow switches. This will not only help in maintaining the required QoS, but minimize the intensity of flooding and data plane saturation attacks.

D. Control-Data Planes Intelligence Tradeoff

In SDN, a single controller or a group of controllers providing control plane services for wider sets of forwarding devices will result in node-to-node or node-to-controller latency in exchanging the network information, and hence open up availability challenges for the logically centralized SDN architectures [51]. Increasing the number of active OpenFlow switches managed by a single controller increases the controller response time for setting the flow rules, since it involves an increased number of flow setup requests [155]. Therefore, intelligence tradeoff between the control and data planes could be one solution to minimize switch dependency on the controller.

Intelligence tradeoff will be beneficial in solving two main challenges in SDNs. First, enhance scalability of the OpenFlow architecture and minimize delays through local decision making capabilities. Second, enhance availability and enable fast restoration. From the network security point of view, sharing intelligence will enable the network to be less prone to single point of failures, as well as less prone to DoS attacks. However, intelligence tradeoff as an attempt to enhance network security has not been demonstrated yet.

E. Synchronization of Network Security and Network Traffic

Network security is an integral part of the network management where stable and robust security policy deployment requires global analysis of policy configuration of all the networked elements to avoid conflicts and inconsistency in the security procedures and hence diminish the chances of serious security breaches and network vulnerabilities [3]. Network security and network traffic challenges are part of the network management where synchronization is needed for cooperative policy deployment. Cooperative policy deployment is needed to keep the network traffic alive even if there is a security breach or the other way around, it keeps the network security intact with changes in the network topology, node mobility and changes in traffic behavior or volumes.

Even though such cooperative policies and procedures have been suggested in various projects, such as 4D [11], Ethane [14], etc. it has never been realized into practice. The main reason behind this lack of cooperation among security and traffic management, as we see today, is the loosely coupled control planes of forwarding devices, independent security architectures and policies, and security independent routing. Hence, Greenberg *et al.* [11] state that traditional IP networks have the characteristics of instability and complexity where a small misconfiguration of a routing protocol can have a severe global impact of cascading meltdown.

In SDNs, since a logically centralized controller is responsible for controlling and managing the whole network, security lapses compromising the controller, nonetheless, will affect setting up flow rules in the data plane. For example, in OpenFlow networks a DoS attack on the centralized controller would at least increase delay in setting up flow rules in the OpenFlow switches. Similar to other networks, in SDNs traffic flow features can be used to detect distributed DoS attacks [64]. However, the global network visibility in the centralized SDN control plane and programmability of the forwarding plane can enable deploying cooperative and interdependent policies. Therefore, interdependent security and traffic forwarding policies, that converge to secure traffic forwarding mechanisms, are needed to take full advantage of SDN technologies.

F. Network Security Automation

The increasing complexity, dynamism, heterogeneity, and requirements of reliability and scalability is making management and monitoring of communication networks much more difficult and prone to errors. However, the challenges of sensing contextual changes in the network, adapting to the contextual changes and control loop for the system to learn and update itself for future actions in traditional networks has staggered deployment of automation techniques [156]. In the context of network security, Hamed *et al.* conclude in [3] that manual configuration of network security technologies, such as firewall and IPSec technologies on extended sets of devices are prone to configuration errors, intra- and inter- policy conflicts resulting in serious security vulnerabilities and threats. A study on firewall configuration errors [4] shows that configuration complexity is one of the main reasons for security breaches in enterprise networks.

SDN abstracts away the low level configuration from network devices and enable designing languages and network controllers which are capable of automatically reacting to the changing network state [157]. ONF (Open Networking Foundation) claims in [158] that SDNs offer flexible network automation and management framework, which makes it possible to develop tools to automate management tasks (that are done manually) to reduce operational overhead, decrease network instability introduced by operator error and support emerging self-services provisioning models. There are automation frameworks for QoS [159], automated policy implementation through Procera [69], and responsive automatic control platform OMNI [160], etc. However, no viable SDN security automation mechanisms have been demonstrated yet.

Centralizing the network control also increases responsibilities on network administrators where the lack of the administrator's knowledge and availability can become a bottleneck for a network. Therefore, automated security mechanisms that require minimal administrator intervention to protect the network and activate automatic recovery mechanisms are needed for SDNs.

G. Identity Location Split

One of the pressing problems pertaining to Internet security is the lack of ensuring proper identities. Jennifer Rexford, a proponent of clean slate Internet architecture, relates security problems that are deeply rooted in the current Internet architecture to the Internet's weak notions of identity [161]. The same legacy remains in SDN where there is no significant work on improving user identity systems in SDN. OpenFlow does provide mechanisms for identifying flows or packets based on packet header fields, but there are no mechanisms that can ensure and bind identities to the users.

The IP address embodies information about the attachment point of the host and works as an identity of the host. This causes semantic collision leading to limited flexibility in the Internet architecture [162]. Using IP addresses in OpenFlow, change of address due to mobility would disrupt flow processing and will require fast and regular updates to flow tables that will cause additional overhead. Therefore, the use of novel technologies such as HIP [143] should be investigated for SDN, to not only provide permanent identities (e.g., Host Identity Tags (HITs) or Host Identifiers (HIs)), but baseline end-to-end security aided by the associated set of security protocols.

X. Conclusion

SDN enhances network security through global visibility of the network state. In SDN, a common distribution layer gathers information about security requirements of different services, resources and hosts, and disseminates security establishing commands to network elements to enforce security policies. On one hand, centralizing the network control plane and enabling network programmability can result in robust and scalable security enforcement, on the other hand, it introduces new security challenges. Therefore, we have presented security weaknesses and strengths of SDN in this paper. In doing so, we have highlighted security vulnerabilities in application, control, and data planes of SDN, and then presented security solutions for these planes. We also summarized security techniques that can strengthen the network-wide security in SDNs. Then we presented security solutions according to the security recommendations of ITU-T and briefly described the costs associated with security solutions.

To our understanding, the most vulnerable component in the SDN architecture is the centralized controller. As a result, controller vulnerability has already been discussed and researched from different perspectives including controllers' safety from applications, controller's scalability and availability, resilience and placement, and security from DoS and DDoS attacks. The security of the OpenFlow switch flow rules have gained more attention of the research community, since they define the actual communication. Although security applications are developed and implemented, the security of the application plane itself is a security challenge. Moreover, communication security between controllers and switches in OpenFlow is threatened due to the optional use of TLS and DTLS.

It is highly possible that new security threats will emerge with the gradual deployment of SDN technologies. Similarly, the threat space will most likely grow, since security threats existing in traditional networks will propagate along with SDN-specific security challenges. However, SDN aims at bringing innovation in communication networks and thus, automatic security mechanisms will be developed to enable fast anomaly detection and quick response for protection. The existing literature on network security in SDN endorses the fact that SDN will enable rapid deployment of cost-effective security services.

Acknowledgment

This work was accomplished in the platform of the CELTIC SIGMONA (SDN Concepts In Generalized Mobile Network Architectures) project. The authors would like to thank the project partners for their thoughtful discussions.

References

[1] B. Raghavan *et al.*, "Software-defined internet architecture: decoupling architecture from infrastructure," in *Proc. 11th ACM Workshop Hot Topics Netw.*, 2012, pp. 43–48.

[2] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[3] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Commun. Mag.*, vol. 44, no. 3, pp. 134–141, Mar. 2006.

[4] A. Wool, "A quantitative study of firewall configuration errors," *Computer*, vol. 37, no. 6, pp. 62–67, Jun. 2004.

[5] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Proc. IEEE SDN4FNS*, 2013, pp. 1–7.

[6] D. Clark, R. Braden, K. Sollins, J. Wroclawski, and D. Katabi, "New Arch: Future generation Internet architecture," Def. Tech. Inf. Center (DTIC), Fort Velvoir, VA, USA, Tech. Rep. AFRL-IF-RS-TR-2004-235, 2004.

[7] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Commun. Mag.*, vol. 35, no. 1, pp. 80–86, Jan. 1997.

[8] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," in *Proc. DARPA Active NEtw. Conf. Expo.*, 2002, pp. 2–15.

[9] Z. Liu, R. Campbell, and M. Mickunas, "Active security support for active networks," *IEEE Trans. Syst., Man, Cybern., Part C, Appl. Rev.*, vol. 33, no. 4, pp. 432–445, Nov. 2003.

[10] S. Murphy, E. Lewis, R. Puga, R. Watson, and R. Yee, "Strong security for active networks," in *Proc. IEEE Open Archit. Netw. Programm.*, 2001, pp. 63–70.

[11] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, Oct. 2005.

[12] Z. Cai, A. L. Cox, and T. E. N. Maestro, "Maestro: A system for scalable OpenFlow control," Rice Univ., Houston, TX, USA, Tech. Rep. TR10-08, 2010.

[13] M. Casado *et al.*, "SANE: A protection architecture for enterprise networks," in *Proc. Usenix Security*, 2006, pp. 137–151.

[14] M. Casado *et al.*, "Ethane: Taking control of the enterprise," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, Oct. 2007.

[15] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1955–1980, 4th Quart. 2014.

[16] S. Namal, I. Ahmad, S. Saud, M. Jokinen, and A. Gurtov, "Implementation of OpenFlow based cognitive radio network architecture: SDN&R," in *Wireless Networks*. New York, NY, USA: Springer, 2015, pp. 1–15.

[17] M. Liyanage, A. Gurtov, and M. Ylianttila, *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. Hoboken, NJ, USA: Wiley, 2015.

[18] S. J. Vaughan-Nichols, "OpenFlow: The next generation of the network?" *Computer*, vol. 44, no. 8, pp. 13–15, Aug. 2011.

[19] T. Nadeau, "Software driven networks problem statement," Network Working Group Internet-Draft, Sep. 30, 2011. [Online]. Available: https://tools.ietf.org/html/draft-nadeau-sdn-problem-statement-00

[20] H. Xie, T. Tsou, D. Lopez, H. Yin, and V. Gurbani, "Use cases for ALTO with software defined networks," Working Draft, IETF Secretariat, Internet-Draft, 2012. [Online]. Available: https://tools.ietf.org/html/draft-xie-alto-sdn-use-cases-01

[21] N. Gude *et al.*, "NOX: towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, Jul. 2008.

[22] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2014.

[23] J. Jeong, H. Kim, and J. Park, "Requirements for security services based on software-defined networking," Internet Eng. Task Force, Fremont, CA, USA, Oct. 2014. [Online]. Available: https://tools.ietf.org/html/draft-jeong-l2nsf-sdn-security-services-00

[24] M. Wasserman and S. Hartman, "Security analysis of the open networking foundation (ONF) OpenFlow switch specification. Internet engineering task force," Internet Eng. Task Force, Fremont, CA, USA, Apr. 2013. [Online]. Available: https://tools.ietf.org/html/draft-mrw-sdnsec-openflow-analysis-02

[25] D. V. Bernardo, "Software-defined networking and network function virtualization security architecture," Internet Eng. Task Force, Fremont, CA, USA. [Online]. Available: https://tools.ietf.org/html/draft-bernardo-sec-arch-sdnnvf-architecture-00

[26] "Security architecture for systems providing end-to-end communications," ITU Telecommun. Standardization Sector, Geneva, Switzerland, 2003.

[27] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, 1st Quart. 2013.

[28] "SDN: Empowering the network," CISCO, San Jose, CA, USA, Nov. 2012. [Online]. Available: http://www.cisco.com/c/en/us/solutions/data-center-virtualization/application-centric-infrastructure/index.html

[29] "VMware NSX virtualization and security platform," VMware, Palo Alto, CA, USA. [Online]. Available: https://www.vmware.com/products/nsx/

[30] "SDN product directory," Open Netw. Found., Palo Alto, CA, USA. [Online]. Available: https://www.opennetworking.org/sdn-resources/onf-products-listing

[31] "SDN market size forecast," SDNCentral, Sunnyvale, CA, USA, Nov. 2013. [Online]. Available: https://www.sdncentral.com

[32] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM Work. Hot Topics Softw. Defined Netw.*, 2013, pp. 55–60.

[33] S. Shin, P. Porras, V. Yegneswaran, and G. Gu, "A framework for integrating security services into software-defined networks," in *Proc. ONS*, 2013, pp. 1–2.

[34] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for OpenFlow applications," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 171–172.

[35] S. Hartman, M. Wasserman and D. Zhang, "Software driven networks problem statement," Network Working Group Internet-Draft, Oct. 15, 2013. [Online]. Available: https://tools.ietf.org/html/draft-hartman-sdnsec-requirements-00

[36] A. D. Ferguson, A. Guha, J. Place, R. Fonseca, and S. Krishnamurthi, "Participatory networking," in *Proc. Hot-ICE*, 2012, pp. 327–338.

[37] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *Proc. 4th ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2008, pp. 1–9.

[38] M. Jarschel *et al.*, "Modeling and performance evaluation of an OpenFlow architecture," in *Proc. 23rd ITCP*, 2011, pp. 1–7.

[39] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proc. CCS*, 2013, pp. 413–424.

[40] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *Proc. 21st IEEE ICNP*, Oct. 2013, pp. 1–2.

[41] S. Shin and G. Gu, "Attacking software-defined networks: a first feasibility study," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 165–166.

[42] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Proc. IEEE NOMS*, Apr. 2012, pp. 933–939.

[43] J. Seedorf and E. Burger, "Application-layer traffic optimization (ALTO) problem statement," Internet Eng. Task Force (IETF), Fremont, CA, USA, 2009.

[44] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated openflow infrastructures," in *Proc. 3rd ACM Workshop SafeConfig*, 2010, pp. 37–44.

[45] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *Proc. IEEE GLOBECOM*, Dec. 2011, pp. 1–6.

[46] T. Dierks, "The Transport Layer Security (TLS) protocol version 1.2," 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246

[47] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," 2012. [Online]. Available: http://tools.ietf.org/html/rfc6347

[48] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proc. 2nd ACM SIGCOMM Workshop HotSDN*, 2013, pp. 151–152.

[49] M. Liyanage and A. Gurtov, "Secured VPN models for LTE backhaul networks," in *Proc. IEEE VTC Fall*, 2012, pp. 1–5.

[50] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements," in *Proc. 18th IEEE Workshop LANMAN*, Oct. 2011, pp. 1–6.

[51] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[52] S. Shin *et al.*, "FRESCO: Modular composable security services for software-defined networks," in *Proc. Netw. Distrib. Security Symp.*, Feb. 2013, pp. 1–16. [Online]. Available: http://www.csl.sri.com/users/vinod/papers/fresco.pdf

[53] R. Beckett *et al.*, "An assertion language for debugging SDN applications," in *Proc3rd ACM Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 91–96.

[54] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model checking invariant security properties in openflow," in *Proc. IEEE ICC*, 2013, pp. 1974–1979.

[55] M. Canini, D. Kostic, J. Rexford, and D. Venzano, "Automating the testing of OpenFlow applications," in *Proc. 1st Int. WRiPE*, 2011, pp. 1–6.

[56] "Security-enhanced floodlight," SDx Central, Sunnyvale, CA, USA. [Online]. Available: http://www.sdncentral.com/education/toward-secure-sdn-control-layer/2013/10/

[57] M. Fernandez, "Comparing openflow controller paradigms scalability: reactive and proactive," in *Proc. IEEE 27th Int. Conf. AINA*, Mar. 2013, pp. 1009–1016.

[58] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE NOMS*, May 2014, pp. 1–4.

[59] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed SDN controllers in a multi-domain environment," in *Proc. IEEE NOMS*, May 2014, pp. 1–2.

[60] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st ACM Workshop HotSDN*, 2012, pp. 7–12.

[61] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *Commun., China*, vol. 11, no. 2, pp. 38–54, Feb. 2014.

[62] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *Proc. IFIP/IEEE Int. Symp. IM*, May 2013, pp. 672–675.

[63] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manag. Conf. Res. Enterprise Netw. USENIX Assoc.*, 2010, p. 3.

[64] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE 35th Conf. LCN*, Oct. 2010, pp. 408–415.

[65] P. Porras *et al.*, "A security enforcement kernel for OpenFlow networks," in *Proc. 1st Workshop HotSDN*, 2012, pp. 121–126.

[66] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 467–472, Sep. 2012.

[67] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proc 1st ACM Workshop Res. Enterprise Netw.*, 2009, pp. 11–18.

[68] N. Foster *et al.*, "Frenetic: A network programming language," *SIGPLAN Notices*, vol. 46, no. 9, pp. 279–291, Sep. 2011.

[69] A. Voellmy, H. Kim, and N. Feamster, "Procera: A language for high-level reactive network control," in *Proc. 1st Workshop HotSDN*, New York, NY, USA, 2012, pp. 43–48.

[70] C. Monsanto, N. Foster, R. Harrison, and D. Walker, "A compiler and run-time system for network programming languages," *SIGPLAN Notices*, vol. 47, no. 1, pp. 217–230, Jan. 2012.

[71] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 55–60.

[72] A. Wundsam, D. Levin, S. Seetharaman, A. Feldmann, "OFRewind: Enabling record and replay troubleshooting for networks." in *Proc. USENIX Annu. Tech. Conf.*, 2011, p. 29.

[73] B. Switch, "Developing floodlight modules. Floodlight OpenFlow controller," 2012. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[74] A. Voellmy and J. Wang, "Scalable software defined network controllers," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 289–290.

[75] "Advanced Message Queuing Protocol." [Online]. Available: http://www.amqp.org

[76] I. Ahmad, S. N. Karunarathna, M. Ylianttila, and A. Gurtov, "Load balancing in software defined mobile networks," in *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. Hoboken, NJ, USA: Wiley, 2015, pp. 225–245.

[77] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "SDN based inter-technology load balancing leveraged by flow admission control," in *Proc. IEEE SDN4FNS*, Nov. 2013, pp. 1–5.

[78] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[79] M. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. Int. CNSM*, Oct. 2013, pp. 18–25.

[80] D. Hock *et al.*, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proc. 25th ITC*, Sep. 2013, pp. 1–9.

[81] J. C. Mogul *et al.*, "DevoFlow: Cost-effective flow management for high performance enterprise networks," in *Proc. 9th ACM SIGCOMM Workshop Hotnets-IX*, 2010, pp. 1–6.

[82] J. Ge, H. Shen, E. Yuepeng, Y. Wu, and J. You, "An OpenFlow-based dynamic path adjustment algorithm for multicast spanning trees," in *Proc. 12th Int. Conf. IEEE TrustCom*, 2013, pp. 1478–1483.

[83] J. Kempf *et al.*, "Scalable fault management for OpenFlow," in *Proc. IEEE ICC*, Jun. 2012, pp. 6606–6610.

[84] E. Maccherani *et al.*, "Extending the NetServ autonomic management capabilities using OpenFlow," in *Proc. IEEE NOMS*, Apr. 2012, pp. 582–585.

[85] F. Hao, T. Lakshman, S. Mukherjee, and H. Song, "Secure cloud computing with a virtualized network infrastructure," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, p. 16.

[86] A. R. Sharafat, S. Das, G. Parulkar, and N. McKeown, "MPLS-TE and MPLS-VPN with OpenFlow," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 452–453, Aug. 2011.

[87] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, Eds. Berlin, Germany: Springer-Verlag, 2011, pp. 161–180.

[88] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using OpenSafe," in *Proc. INM/WREN*, 2010, p. 8.

[89] Z. A. Qazi *et al.*, "SIMPLE-fying middlebox policy enforcement using SDN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, Oct. 2013.

[90] M. Liyanage *et al.*, "Security for future software defined mobile networks," in *Proc. 9th Int. Conf. NGMAST*, 2015, pp. 1–9.

[91] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proc. 1st Workshop HotSDN*, 2012, pp. 79–84.

[92] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker, "Expressing and enforcing flow-based network security policies," Univ. Chicago, Chicago, IL, USA, Tech. Rep., 2009, pp. 1–20. [Online]. Available: http://www.cs.uic.edu/~hinrichs/papers/hinrichs2008design.pdf

[93] R. Bifulco and F. Schneider, "OpenFlow rules interactions: Definition and detection," in *Proc. IEEE SDN4FNS*, Nov. 2013, pp. 1–6.

[94] S. Shirali-Shahreza and Y. Ganjali, "Empowering software defined network controller with packet-level information," in *Proc. IEEE ICC*, Jun. 2013, pp. 1335–1339.

[95] S. Shirali-Shahreza and Y. Ganjali, "Efficient implementation of security applications in openflow controller with flexam," in *Proc. IEEE 21st Annu. Symp. HOTI*, Aug. 2013, pp. 49–54.

[96] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Proc. IEEE ICACT*, Feb. 2014, pp. 167–171.

[97] D. Erickson, "The Beacon Openflow Controller," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, Hong Kong, China, 2011, pp. 13–18. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491189

[98] R. Skowyra, S. Bahargam, and A. Bestavros, "Software-Defined IDS for securing embedded mobile devices," in *Proc. IEEE HPEC*, Sep. 2013, pp. 1–7.

[99] L. Zhang, G. Shou, Y. Hu, and Z. Guo, "Deployment of intrusion prevention system based on software defined networking," in *Proc. IEEE ICCT*, Nov. 2013, pp. 26–31.

[100] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "SnortFlow: A openflow-based intrusion prevention system in cloud environment," in *Proc. 2nd GREE*, Mar. 2013, pp. 89–92.

[101] H. Li and D. Liu, "Research on intelligent intrusion prevention system based on Snort," in *Proc. Int. Conf. CMCE*, Aug., vol. 1 2010, pp. 251–253.

[102] M. Roesch, "Snort: Lightweight intrusion detection for networks." in *Proc. LISA*, 1999, vol. 99, pp. 229–238.

[103] M. Femminella, R. Francescangeli, G. Reali, J. W. Lee, and H. Schulzrinne, "An enabling platform for autonomic management of the future internet," *IEEE Netw.*, vol. 25, no. 6, pp. 24–32, Nov. 2011.

[104] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in SDN," in *Proc. IEEE MILCOM*, Nov. 2013, pp. 992–997.

[105] N. L. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow software-defined networks," in *Proc. IEEE NOMS*, May 2014, pp. 1–8.

[106] X. Liu, H. Xue, X. Feng, and Y. Dai, "Design of the multi-level security network switch system which restricts covert channel," in *Proc. IEEE 3rd ICCSN*, May 2011, pp. 233–237.

[107] S. R. Chowdhury, M. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low cost network monitoring framework for software defined networks," in *Proc. IEEE NOMS*, May 2014, pp. 1–9.

[108] C.-T. Yang *et al.*, "Implementation of a virtual switch monitor system using openflow on cloud," in *Proc. Int. Conf. IMIS Ubiquitous Comput.*, Jul. 2013, pp. 283–290.

[109] "OpenFlow Firewall: A Floodlight Module. " [Online]. Available: http://www.openflowhub.org/display/floodlightcontroller

[110] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: building robust firewalls for software-defined networks," in *Proc. 3rd Workshop Topics Softw. Defined Netw.*, 2014, pp. 97–102.

[111] M. Koerner and O. Kao, "Oftables: A distributed packet filter," in *Proc. 6th Int. Conf. COMSNETS*, Jan. 2014, pp. 1–4.

[112] M. Suh, S. H. Park, B. Lee, and S. Yang, "Building firewall over the software-defined network controller," in *Proc. 16th ICACT*, Feb. 2014, pp. 744–748.

[113] J. P. Sterbenz *et al.*, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Comput. Netw.*, vol. 54, no. 8, pp. 1245–1265, 2010.

[114] P. Smith, A. Schaeffer-Filho, D. Hutchison, and A. Mauthe, "Management patterns: SDN-enabled network resilience management," in *Proc. IEEE NOMS*, May 2014, pp. 1–9.

[115] K. Nguyen, Q. T. Minh, and S. Yamada, "A software-defined networking approach for disaster-resilient WANs," in *Proc. 22nd ICCCN*, Jul. 2013, pp. 1–5.

[116] K. Nguyen, Q. T. Minh, and S. Yamada, "Towards optimal disaster recovery in backbone networks," in *Proc. IEEE 37th Annu. COMPSAC*, 2013, pp. 826–827.

[117] D. A. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," in *Proc. ACM SIGCOMM*, New York, NY, USA, 2008, pp. 51–62.

[118] J. Lee, J. Tourrilhes, P. Sharma, and S. Banerjee, "No more middlebox: Integrate processing into network," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 459–460, Aug. 2010.

[119] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward Software-defined Middlebox Networking," in *Proc. 11th ACM Workshop HotNets-XI*, 2012, pp. 7–12.

[120] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, "A slick control plane for network middleboxes," in *Proc. 2nd ACM SIGCOMM Workshop HotSDN*, 2013, pp. 147–148.

[121] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "FlowTags: Enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proc. 2nd ACM SIGCOMM Workshop HotSDN*, 2013, pp. 19–24.

[122] Y.-J. Chen, F.-Y. Lin, L.-C. Wang, and B.-S. Lin, "A dynamic security traversal mechanism for providing deterministic delay guarantee in SDN," in *Proc. IEEE Int. Symp. WoWMoM*, Jun. 2014, pp. 1–6.

[123] R. Hand, M. Ton, and E. Keller, "Active security," in *Proc. 12th ACM Workshop HotNets-XII*, 2013, pp. 17:1–17:7.

[124] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions," in *Proc. IEEE NOMS*, May 2014, pp. 1–9.

[125] K. Wang, Y. Qi, B. Yang, Y. Xue, and J. Li, "LiveSec: Towards effective security management in large-scale production networks," in *Proc. ICDCSW*, Jun. 2012, pp. 451–460.

[126] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, “Virtualizing the network forwarding plane,” in *Proc. Workshop Programm. Routers Extensible Serv. Tomorrow*, 2010, p. 8.

[127] J. Costa-Requena *et al.*, “SDN and NFV integration in generalized mobile network architecture,” in *Proc. EuCNC*, Jun. 2015, pp. 154–158.

[128] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, “Towards software defined cognitive networking,” in *Proc. Int. Conf. NTMS*, 2015, pp. 1–5.

[129] A. van Cleeff, W. Pieters, and R. Wieringa, “Security implications of virtualization: A literature study,” in *Proc. Int. Conf. CSE*, Aug. 2009, vol. 3, pp. 353–358.

[130] S. Vaughan-Nichols, “Virtualization sparks security concerns,” *Computer*, vol. 41, no. 8, pp. 13–15, Aug. 2008.

[131] R. Sherwood *et al.*, “Flowvisor: A network virtualization layer,” OpenFlow Switch Consortium, Tech. Rep. OPENFLOW-TR-2009-1, Stanford Univ., Stanford, CA, USA, 2009.

[132] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, “NICE: Network intrusion detection and countermeasure selection in virtual network systems,” *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 4, pp. 198–211, Jul. 2013.

[133] “SDN security considerations in the data center. Open networking foundation,” Open Netw. Found., Palo Alto, CA, USA. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-library

[134] S. Shin and G. Gu, “CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?),” in *Proc. 20th IEEE ICNP*, Oct. 2012, pp. 1–6.

[135] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, “A novel design for future on-demand service and security,” in *Proc. 12th IEEE ICCT*, 2010, pp. 385–388.

[136] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, “Enabling secure mobility with OpenFlow,” in *Proc. IEEE SDN4FNS*, 2013, pp. 1–5.

[137] G. Yao, J. Bi, and P. Xiao, “Source address validation solution with OpenFlow/NOX architecture,” in *Proc. 19th IEEE ICNP*, 2011, pp. 7–12.

[138] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “OpenFlow random host mutation: transparent moving target defense using software defined networking,” in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 127–132.

[139] M. A. S. Santos *et al.*, “Software-defined networking based capacity sharing in hybrid networks,” in *Proc. IEEE ICNP*, 2013, pp. 1–6.

[140] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, and N. Zeldovich, “Delegating network security with more information,” in *Proc. 1st ACM Workshop Res. Enterprise Netw.*, 2009, pp. 19–26.

[141] Z. Faigl, J. Pellikka, L. Bokor, and A. Gurtov, “Performance evaluation of current and emerging authentication schemes for future 3GPP network architectures,” *Comput. Netw.*, vol. 60, pp. 60–74, 2014.

[142] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, “The locator/ID separation protocol (LISP),” CISCO, San Jose, CA, USA, 2013.

[143] A. Gurtov, *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Hoboken, NJ, USA: Wiley, 2008.

[144] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, “Effective flow protection in OpenFlow rings,” in *Proc. OFC/NFOEC*, Mar. 2013, pp. 1–3.

[145] J. Li, J. Hyun, J.-H. Yoo, S. Baik, and J.-K. Hong, “Scalable failover method for data center networks using OpenFlow,” in *Proc. IEEE NOMS*, May 2014, pp. 1–6.

[146] P. Nikander, A. Gurtov, and T. Henderson, “Host Identity Protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks,” *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 186–204, 2nd Quart. 2010.

[147] M. Liyanage, M. Ylianttila, and A. Gurtov, “Securing the control channel of software-defined mobile networks,” in *Proc. IEEE 15th Int. Symp. WoWMoM*, Jun. 2014, pp. 1–6.

[148] A. R. Curtis *et al.*, “DevoFlow: Scaling flow management for high-performance networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.

[149] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, “Scalable flow-based networking with DIFANE,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 351–362, Oct. 2011.

[150] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, “Enabling fast failure recovery in OpenFlow networks,” in *Proc. 8th Int. Workshop DRCN*, 2011, pp. 164–171.

[151] L. Vanbever, J. Reich, T. Benson, N. Foster, and J. Rexford, “HotSwap: Correct and efficient controller upgrades for software-defined networks,” in *Proc. 2nd ACM SIGCOMM HotSDN*, 2013, pp. 133–138.

[152] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, “Extending networking into the virtualization layer,” in *Proc. Hotnets*, 2009, pp. 1–6.

[153] A. Gember, C. Dragga, and A. Akella, “ECOS: Leveraging software-defined networks to support mobile application offloading,” in *Proc. 8th ACM/IEEE Symp. ANCS*, 2012, pp. 199–210.

[154] A. Keromytis, “Voice-over-IP security: Research and practice,” *IEEE Security Privacy*, vol. 8, no. 2, pp. 76–78, Mar. 2010.

[155] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, “On controller performance in software-defined networks,” in *Proc. USENIX Workshop Hot-ICE*, 2012, p. 10.

[156] B. Jennings *et al.*, “Towards autonomic management of communications networks,” *IEEE Commun. Mag.*, vol. 45, no. 10, pp. 112–121, Oct. 2007.

[157] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[158] “Software-defined networking: The new norm for networks,” Open Netw. Found., Palo Alto, CA, USA. [Online] Available: https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers/benefits-of-OFB-SDN

[159] W. Kim *et al.*, “Automated and scalable QoS control for network convergence,” in *Proc. INM/WREN*, 2010, vol. 10, pp. 1–1.

[160] D. M. Mattos *et al.*, “OMNI: OpenFlow management infrastructure,” in *Proc. IEEE Int. Conf. NOF*, 2011, pp. 52–56.

[161] J. Rexford and C. Dovrolis, “Future internet architecture: clean-slate versus evolutionary research,” *Commun. ACM*, vol. 53, no. 9, pp. 36–40, Sep. 2010.

[162] T. Li, “Design goals for scalable internet routing,” ip.com, Fairport, NY, USA, 2011.

Ijaz Ahmad received the M.Sc. (Technology) degree in wireless communications engineering with major in telecommunication engineering from University of Oulu, Finland, in 2012. He is pursuing the doctoral degree with the Department of Communications Engineering, University of Oulu, and a Research Scientist with the Centre for Wireless Communications (CWC), Oulu, Finland. His research interest includes SDN, SDN security, software defined mobile networks, OpenFlow, and network load balancing.

Suneth Namal received the B.Sc. degree in computer engineering from the University of Peradeniya, Peradeniya, Sri Lanka, in 2007. He completed the M.E. degree in information and communication technologies from the Asian Institute of Technology, Bangkok, Thailand, in 2010 and M.Sc. degree in communication network and services from Telecom Sud-Paris, Paris, France, in 2010. He is pursuing the doctoral degree with the Department of Communication Engineering, University of Oulu, Oulu, Finland. His research interest includes software defined networks (SDN), OpenFlow, network security, mobile femtocells, fast initial authentication, load balancing and mobility management, Wi-Fi protocols. Currently, he is a Senior Lecturer at the Department of Computer Engineering, University of Peradeniya, Sri Lanka.

Mika Ylianttila (M'99–SM'08) received the doctoral degree in communications engineering at the University of Oulu, Finland, in 2005, and he has worked as a Researcher and Professor since. He is the director of the Center for Internet Excellence (CIE) Research and Innovation Unit, and part-time Professor with the Department of Communications Engineering and Centre for Wireless Communications. He is also Adjunct Professor (docent) in the Department of Computer Science and Engineering. He has co-authored about 100 peer-reviewed articles on networking, decentralized systems, mobility management, and content distribution. Based on Google Scholar, his research has impacted more than 1800 citations and h-index is 19. He was a visiting researcher at Center for Wireless Information Network Studies (CWINS), Worcester Polytechnic Institute, Massachusetts, and Internet Real Time Lab (IRT), Columbia University, New York, NY, USA. He is the Editor of the journal, *Wireless Networks* (Springer).

Andrei Gurtov (S'98–A'00–M'05–SM'10) received the M.Sc. and Ph.D. degrees in computer science from the University of Helsinki, Helsinki, Finland, in 2000 and 2004, respectively. He is presently a Principal Scientist at the Helsinki Institute for Information Technology (HIIT). He is also Adjunct Professor at Aalto University, University of Helsinki, and University of Oulu. He was a Professor at University of Oulu in the area of wireless Internet in 2010–2012. Previously, he worked at TeliaSonera, Ericsson NomadicLab, and the University of Helsinki. He was a Visiting Scholar at the International Computer Science Institute (ICSI), Berkeley in 2003, 2005, and 2013. Dr. Gurtov is a co-author of over 150 publications including three books, research papers, patents, and five IETF RFCs. He is a senior member and ACM.