SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:5803–5833
Published online 7 February 2017 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1737

RESEARCH ARTICLE

Software-defined networking (SDN): a survey

Kamal Benzekki*, Abdeslam El Fergougui and Abdelbaki Elbelrhiti Elalaoui

Laboratory of Computer Networks and Systems, Department of Mathematics and Computer Science, Faculty of Sciences, Moulay Ismail University, Meknes, Morocco

ABSTRACT

With the advent of cloud computing, many new networking concepts have been introduced to simplify network management and bring innovation through network programmability. The emergence of the software-defined networking (SDN) paradigm is one of these adopted concepts in the cloud model so as to eliminate the network infrastructure maintenance processes and guarantee easy management. In this fashion, SDN offers real-time performance and responds to high availability requirements. However, this new emerging paradigm has been facing many technological hurdles; some of them are inherent, while others are inherited from existing adopted technologies. In this paper, our purpose is to shed light on SDN related issues and give insight into the challenges facing the future of this revolutionary network model, from both protocol and architecture perspectives. Additionally, we aim to present different existing solutions and mitigation techniques that address SDN scalability, elasticity, dependability, reliability, high availability, resiliency, security, and performance concerns. Copyright © 2017 John Wiley & Sons, Ltd.

KEYWORDS
software-defined networking; SDN challenges; SDN issues; SDN architecture; OpenFlow

***Correspondence**
Kamal Benzekki, Laboratory of Computer Networks and Systems, Department of Mathematics and Computer Science, Faculty of Sciences, Moulay Ismail University, Meknes, Morocco.
E-mail: benzekki@gmail.com

1. INTRODUCTION

Software-defined networking (SDN) is facilitating organizations to deploy applications and enable flexible delivery, offering the capability of scaling network resources in lockstep with application and data needs, and reducing both CapEX and OpEX [1]. SDN is an innovative approach to design, implement, and manage networks that separate the network control (control plane) and the forwarding process (data plane) for a better user experience. This network segmentation offers numerous benefits in terms of network flexibility and controllability. In the one hand, it allows combining the advantages of system virtualization and cloud computing and on the other hand, to create an implementation of a centralized intelligence that enables making a clear visibility over the network for the sake of easy network management and maintenance as well as enhanced network control and reactivity. In the conventional infrastructure (Figure 1 and Table I), network implementation, configuration, and troubleshooting require high-level technically skilled network and system engineer intervention and operational costs involved in provisioning and managing large multivendor networks. In fact, the variety and the complexity [2] of network elements make their maintenance very expensive and the underlying infrastructure less reliable in case of frequent network failures, especially if no backup plans are anticipated within the infrastructure.

As SDN separates the routing and forwarding decisions of networking elements (e.g., routers, switches, and access points) from the data plane, the network administration and management become uncomplicated because the control plane only deals with the information related to logical network topology, the routing of traffic, and so on. In contrast, the data plane orchestrates the network traffic in accordance with the established configuration in the control plane.

In SDN, the control operations are centralized in a controller that dictates the network policies. Many controller platforms are open source such as Floodlight [3], OpenDayLight [4], and Beacon [5]. The management of the network can be achieved on different layers (i.e., application, control, and data plane). For instance, service providers can allocate resources to customers via application layer, configure and modify network policies and logical entities on control plane, and set up physical network elements on data plane.

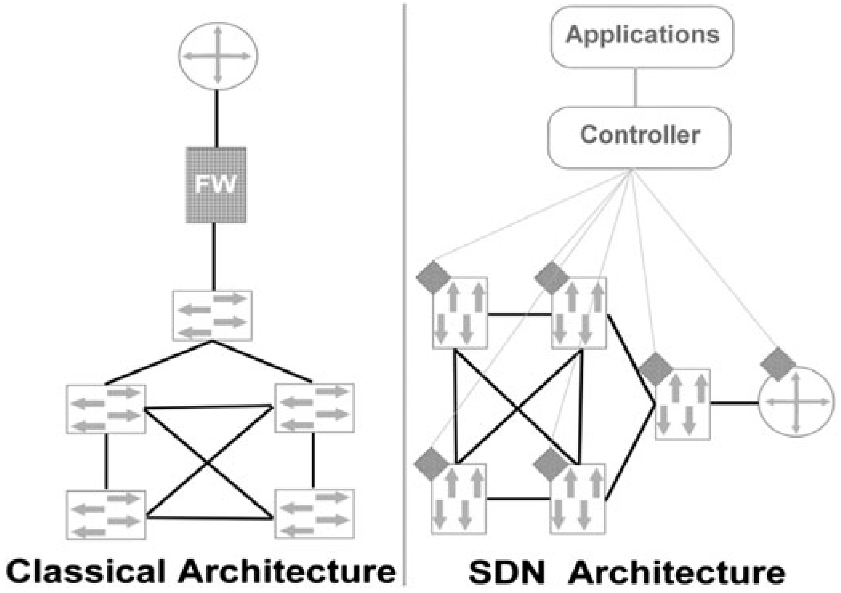

Figure 1. Software-defined networking (SDN) versus classical architecture.

Software-defined networking has come to light in recent years. However, the concept of this approach has been evolving since the mid-1990s. Ethane [6] (management architecture) and OpenFlow [7] (protocol for network flow) have given birth to a real implementation of SDN. OpenFlow is a protocol that provides a standardized way of managing traffic and describes how a controller communicates with network devices like switches and routers. The devices supporting OpenFlow consist of two logical components: a flow table that defines how to process and forward packets within the network and an exposed OpenFlow application programming interface (API) that handles the exchanges between switch/router and controller.

OpenFlow is standardized by the Open Networking Foundation [8]. There are other similar existing southbound interfaces such as the Forwarding and Control Element Separation [9,10] standardized by the Internet Engineering Task Force, Path Computation Element [11], the Locator/ID Separation Protocol [12], and SoftRouter [13].

Table I. Software-defined networking versus classical networking.

Characteristics	SDN architecture	Classical architecture
Programmability	✓	
Centralized control	✓	
Error-prone configuration		✓
Complex network control		✓
Network flexibility	✓	
Improved performance	✓	
Easy implementation	✓	
Efficient configuration	✓	
Enhanced management	✓	

Furthermore, the demand for cloud services in its various forms (e.g., SaaS, PaaS, IaaS, Network-as-a-Service [NaaS], DaaS, UCaaS, etc.) is increasing drastically. Although these services are centralized in data centers, they pose important challenges for service providers [14]. With the rapid growth of the clients' demands, the operator is required to respond accordingly by considering additional servers, network components, high quality of service, and secure architecture [15] abiding by the standards. This generally comes first at the cost of non-negligible effort in facing new challenges appearing within the core network where SDN leads and governs. In particular, the challenges and issues that appear of paramount importance in the SDN environment are the following:

(1) *Scalability*: This defines the ability of SDN to, more specifically in the control plane, handle and process an increasing workload. Scalability aims at enlarging the capacity of the SDN by implementing mechanisms such as devolving [16], clustering [17], and high processing [18] to cope with the growing load.
(2) *Reliability*: The SDN is considered reliable when it notifies of delivery failures of the data in real time. In such network, there should be a specified minimum reliability for the delivery of critical data. In today's implementations, SDN controllers [19] must be capable of meeting real-time requirements for reliable delivery and timeliness.
(3) *High availability*: HA is an important aspect of today's services that should be available each time a customer requests a given service or resource.

Availability is usually expressed as a percentage of uptime in a given year. Unavailability of services can generally occur owing to network outages or system crashes [20,21]. Network providers generally deploy backup services to offer HA by implementing redundant server hardware, server OS and network components, and so on.

(4) *Elasticity*: Elasticity is the ability of SDN to dynamically adapt its capacity by scaling up or down the available resources [22,23] in order to meet the variation and fluctuation of the workload. Generally, elasticity is often focused on the control plane and might be referred to as scalability.

(5) *Security*: SDN security consists of protecting the information from theft or damage to the hardware and the software as well as from disruption of the services [24,25]. Securing SDN encompasses physical security of the hardware, as well as preventing logical threats that may come from the network or data. SDN vulnerabilities are the entrance door to security attacks being intentional or accidental.

(6) *Performance*: Performance refers to the amount of tasks achieved by SDN components compared with the time/resources (e.g., CPU and RAM) being used [26]. There are many different ways to measure [27,28] the performance of a network, as each network is different in nature and design. As far as SDN is concerned, the important measures are bandwidth, throughput, latency, and jitter.

(7) *Resilience*: Resilience in SDN is the capability to ensure and maintain an acceptable level of service even in case of a service, network or node failure. When an SDN element is faulty, the network should provide a continuous operational service with the same performance. In order to increase the resilience of SDN, potential challenges/risks have to be identified and addressed to protect the services [29].

(8) *Dependability*: The dependability of SDN is strongly tied to availability and reliability terms. SDN dependability aims mainly at preventing faults and implementing fault tolerance mechanisms to guarantee service delivery even at a degraded level [30]. In addition to HA and reliability, integrity and maintainability are also two important dependability attributes.

The rest of this paper is organized as follows. Section 2 gives an overview of SDN technology and describes the generic architecture and the adopted communication protocols. Section 3 discusses scalability and elasticity challenges/issues facing SDN in current deployments as well as the existing solutions to meet SLA [31] requirements. Section 4 introduces dependability, HA, and reliability and analyzes their challenges/issues, impacts, and workarounds to satisfy the customer and abide by the depicted specifications in the SLA. SDN performance is discussed and explored with the research efforts made so far to improve network responsiveness and minimize delays in Section 5. Section 6 gives insight into how to achieve SDN resilience and presents existing works in this sense. Security threats, vulnerabilities, and mitigation techniques are provided in Section 7. Finally, Section 8 summarizes and concludes this paper.

2. SOFTWARE-DEFINED NETWORKING: AN OVERVIEW

2.1. Generic software-defined networking architecture

Unlike conventional IP networks (Figure 1) whose functionalities are decentralized, SDN is centralized to offer connection network domains between the control and data plane in the same infrastructure. Additionally, SDN allows backward compatibility with existing protocols and standards (e.g., IP, ARP, VLAN, Ethernet, etc.) [6].

In Figure 2, the core architecture of SDN is divided into three layers. The upper layer of SDN architecture is an application layer that defines rules and offers different services such as firewall, access control, IDS/IPS, quality of service, routing, proxy service, and monitoring balancer. This layer is responsible of abstracting the SDN network control management through the northbound API (e.g., OpenDaylight) [4]. The second layer is known as the control plane, which is an abstraction of the network topology. The controller is the main component responsible for establishing flow tables and data handling policies as well as abstracting the network complexity and collecting network information through the southbound API and maintaining an up-to-date network holistic view. The southbound API communications can be deployed in two different scenarios:

- in-band communication: In this scenario, the traffic between the controller and any network device should abide by the dictated flow rules.
- out-of-band communication: Here, the traffic does not follow flow rules. This type of deployment requires VLAN implementation to isolate the traffic flow from the communications, which depends on OpenFlow rules.

There are also eastbound/westbound APIs (e.g., HyperFlow) [17] that enable multiple controllers to exchange control information regarding the flow in the data plane. We can find several existing controllers based on different programming languages (Python, C/C++, Java, Ruby, etc.) and platforms such as NOX [32], Floodlight [3], Beacon [5], Maestro [33], and Trema [34]. The lowest layer, which is known as the data plane, provides networking devices such as physical/virtual switches, routers, and access points and is responsible for all data activities including forwarding, fragmentation, and reassembly.

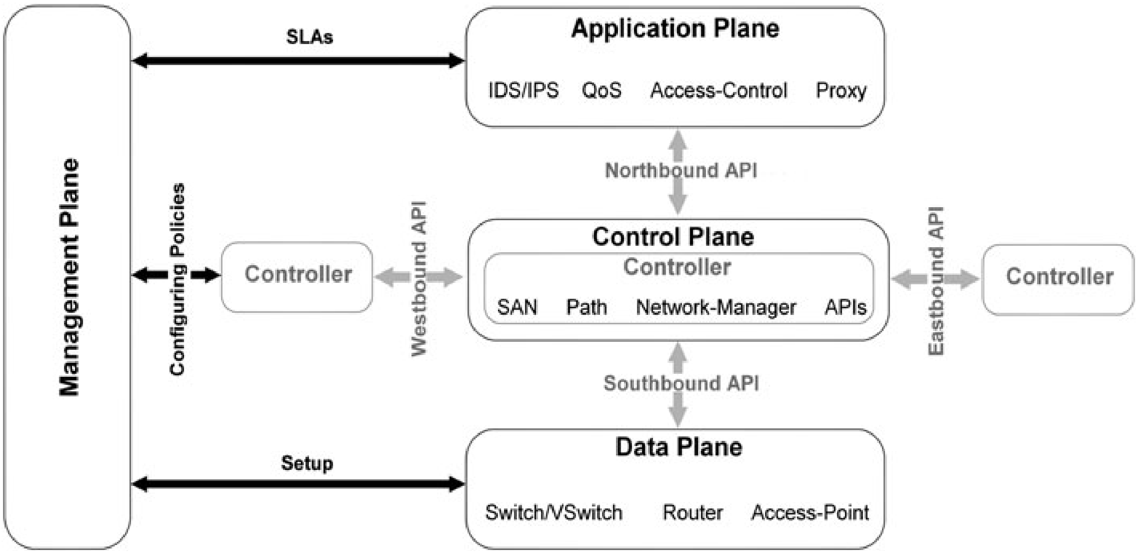

Figure 2. A generic software-defined networking architecture.

2.2. Existing network cloud-based models

There are multiple cloud-based models solutions for enhancing network functionalities and facilitating network management of the underlying infrastructure like in NaaS (Table II). For instance, an OpenFlow-based SDN model [7] can be adopted to support [35] NaaS, enhance performance, and enforce the global visibility of the network. Additionally, a wide array of reliable applications is offered to the users to take benefit from advanced network functions. There are also two other cloud-based models: network virtualization model [36,37] and evolutionary model [38]. The network virtualization model can be applied to SDN to provide scalability in VLAN. This approach allows creating several virtual network instances in a single physical infrastructure. This model has been introduced to resolve issues related to mobility across subnets and isolation in joint-tenant environments. The evolutionary model aims at dividing the network into virtual segments and offers quality of service, fault tolerance and configuration management, and so on.

Table II. Cloud-based models.

	Description	Capabilities and features	Limitations
OpenFlow-based SDN model [7]	First designed standard interface communication protocol for SDN controllers to allow programming the forwarding tables of the network elements separation/centralization/programmability of the control plane	Simplifying network operations Improving network availability Providing better control and performance for SDN	OF switches support issues Compatibility issues with controller and OF switches
Network virtualization model [36,37]	Coexisting multiple network instances on a common physical network to facilitate network provisioning	Addressing the scalability issues with multicasting VLAN architectures Offering multi-tenancy to the cloud	An implementation of deep packet inspection mechanism is required to identify the individual virtual network's header by the network nodes
Evolutionary model [38]	Partitioning the network into virtual communities to manage traffic and quality of service on the basis of network standards (e. g., MPLS, VXLAN) by using management interfaces	Enhancing software control of the network Ensuring flexible virtual networks extension	Existing interoperability issues between vendors that may not consider the implementation of some standards/protocols (Some existing solutions like OpenStack can help enabling interoperability of network devices.)

SDN, software-defined networking.

DOI: 10.1002/sec

2.3. Communication protocol

The most common southbound interface in SDN relies on OpenFlow. The OpenFlow specification defines the protocol that enables the controller to command switches and routers.

There exist three communication ways in the OpenFlow protocol: controller-to-switch, asynchronous, and symmetric communication. The controller-to-switch communication is responsible for establishing handshakes, switch, and flow table configuration. The asynchronous communication starts from OpenFlow-compliant switch to inform the controller by sending packet-in message, port status message, and flow-removed message (Figure 3a). For the symmetric communication, there is no restriction about who initiates the exchanges. Examples of exchanged messages are Hello and Echo that are used to aid in detecting problems in the switch-controller connection; other messages are Feature Request/Reply, Set Config, and so on (Figure 3b).

The communication between the controller and the switch can be either run over Transport Layer Security (TLS) or without an encryption mechanism, depending on the security needs and deployments.

3. SCALABILITY AND ELASTICITY

3.1. Scalability challenges and solutions

As the SDN design enables pushing all the control functionality to a centralized controller with a complete network-wide view, developing control applications and enforcing policies become much easier with these properties. However, controllers might potentially become a bottleneck for network operations. As the size of the network grows, more events and requests are sent to the controller, and at some point, handling all the incoming requests by the controller becomes a serious challenge. This is a common problem that exists in any system and does not only apply to SDN design. By decoupling the control and data planes and entrusting the controller for the control of the network traffic over a large network, the growth of network traffic may not scale with the capabilities of the controller in terms of performance and thus engendering scalability and elasticity issues [22], [23]. Multiple works have been proposed to alleviate the workload on one single controller by distributing and sharing tasks across multiple controllers. These are only workarounds to solve specific concerns of SDNs overhead on the control plane and latencies. These scalability solution proposals include DIFANE [16], DevoFlow [39], software-defined counters [40], Onix [41], Kandoo [42], HyperFlow [17], Maestro [33] and NOX-MT [43], which mainly aim at reducing the overhead of the control plane by delegating some work to the forwarding devices or trying to increase the throughput and latency of the controllers by implementing high-performance computing techniques such as buffering, pipelining, multithreading and parallelism. Several works have been proposed to mitigate the scalability issue in the control plane, and few researches are dealing with the data plane [44–46]. From another perspective, these works can be divided into three types, depending on the mechanisms used to achieve scalability: controller capacity improvement (e.g., high-processing techniques), controllers' clusters (e.g., distributed event-based control plane), and devolving/delegating some management functions to the data plane. Table III presents a non-exhaustive list of the proposed solutions to address scalability issues.

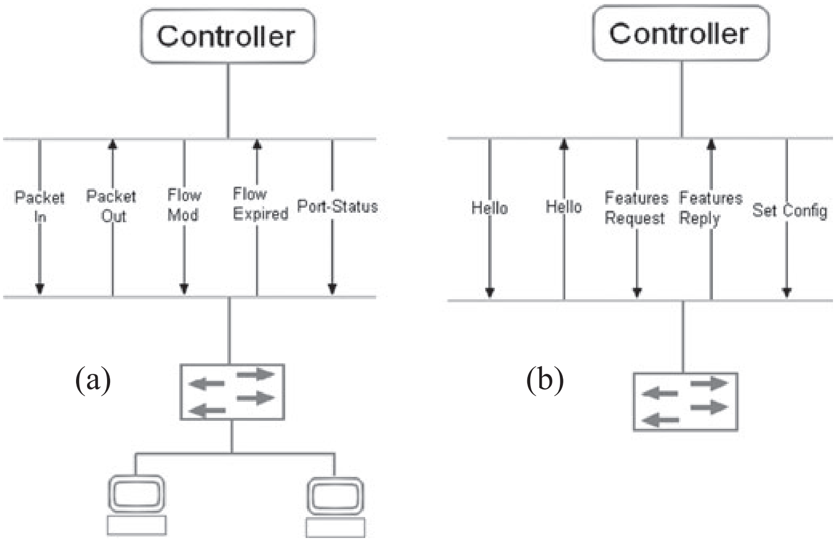

Figure 3. OpenFlow exchanges.

Table III. Scalability issues, solutions, and challenges.

Reference	Proposal	Main purpose/challenge	Scalability (technique)
Tootoonchian *et al*. [17]	HyperFlow localizes decision making to individual controllers in order to minimize the control plane response time to data plane requests	By logically centralizing and physically distributing the network overhead across the control plane, Hyperflow provides scalability for SDN	Clustering
Koponen *et al*. [41]	ONIX platform on top of which a network control plane can be implemented as a distributed system for large-scale networks provides more general APIs for control plane implementations	Onix allows trade-offs between scalability and consistency/durability	Clustering
Berde et al. [85]	Open Network Operating System adopts a distributed architecture for high availability and scale-out. It provides a global network view to applications, which is logically centralized even though it is physically distributed across multiple servers	Improving scalability and performance to meet the requirements of large-scale networks	Clustering
Krishnamurthy *et al*. [51]	Pratyaastha is a new way to partition SDN application state that considers the dependencies between application state and SDN switches	Improving the performance of the current distributed SDN control platforms by proposing a novel approach for assigning SDN switches and partitions of SDN application state to distributed controller instances	Clustering
Yeganeh and Ganjali [42]	Kandoo is a framework that allows scalability preservation without changing switches' organization. Kandoo consists of two layers: a bottom layer grouping controllers with no knowledge of the network-wide state and a top layer containing a logically centralized controller that maintains the network-wide state	Offloading the control applications by separating the controllers into two different roles such as a root controller and a local controller; in this way, the root controller processes rare events and the overhead will be pushed to the local controller in order to provide scalability	Clustering
Veisllari *et al*. [47]	Investigating the scalability of the SDN controller with respect to performance. Scalability is related to performance when we consider metrics such as delay, latency, and throughput	The current Internet flow definitions have high requirements on the processing rate of the SDN controller	Clustering
Park *et al*. [48]	RAON implementation consists of two parts: a RAON switch and a RAON OpenFlow controller abstraction module that communicates with the RAON switch. The networks of the lower-level controllers are abstracted as big OpenFlow switches	Solving scalability issues in OpenFlow networks by reducing complexity and increasing manageability	Clustering
Yu *et al*. [16]	DIFANE is a distributed flow management architecture that runs a partitioning algorithm to divide the rules over switches that will handle flows instead of the controller	Offering a scalable and efficient solution that handles the flow in the data plane by forwarding packets through intermediate switches that maintain necessary rules	Devolving
Curtis *et al*. [39]	DevoFlow aims at reducing the interactions between OF switches and controller using filtering and sampling techniques such as rule aggregation, selective local action, and approximating	Maintaining a global visibility and providing a scalable flow management simulations for large-scale networks	Devolving
Luo *et al*. [45]	Control-message quenching is a mechanism whereby a control-message quenching switch sends only one packet-in message for each source-destination pair, which reduces unnecessary packet-in messages from the OpenFlow switch to the controller	Enhancing the controller responsiveness and network scalability with reduced flow setup latency and higher network throughput	Devolving
Kempf *et al*. [46]	Devolving fault management to the OpenFlow switches by extending the OpenFlow protocol to support the monitoring function. OF switch emits monitoring messages without posing a processing load on the controller	Resolving scalability limitation of the centralized fault management that is based on LLDP messages for monitoring	Devolving

(*Continues*)

Table III. (Continued)

Reference	Proposal	Main purpose/challenge	Scalability (technique)
Benzekki *et al.* [49]	Devolving IEEE 802.1X authentication capability to data plane by inserting a legacy switch that supports IEEE 802.1X port-based authentication and placing the RADIUS server and other servers near the legacy switch to reduce authentication delays	Improving access control and performance and reducing the high demand on the SDN controller when users attempt to authenticate against a RADIUS server	Devolving
Cai *et al.* [18,100]	Maestro is a controller that has a task manager that manages incoming computations and distributes the processing to each controller instance at each core of the processor. Maestro exploits more additional throughput optimization techniques	Improving the capacity of the controller itself by using multicores with parallel processing and multithreads for a scalable SDN	High-processing
Voellmy *et al.* [50]	McNettle is an extensible SDN control system implemented in Haskell that leverages the multicore facilities of the Glasgow Haskell Compiler and runtime system.	Scheduling event handlers, allocating memory, and reducing system calls in order to optimize cache usage and avoids contention on sockets and ensures that each message is processed on a single CPU core	High-processing

SDN, software-defined networking.

3.2. Elasticity

Elasticity is another challenge facing cloud providers to manage and adapt responsiveness of SDN demands in real time. In addition to this, cloud providers should offer more elastic access to computing and network resources. Elasticity can be seen from three dimensions as illustrated in Figure 4.

(1) elastic scalability: aims at ensuring the growth depending on network requirements. SDN can scale dynamically in case of high demands.
(2) elastic configuration: gives an opportunity to add, edit, and delete configuration to meet new network requirements and respond to new challenges in terms of bandwidth and throughput.
(3) elastic discovery*:* anticipates discoveries as well as becoming aware of possible emerging technological opportunities and threats at the right time to react and meet the new requirements and trends. The main purpose is to identify irrelevant tendencies and think of acquiring new efficient techniques/strategies that might help in improving the SDN. The observations should be made quickly and the action must be taken in real time to take advantage of new possible network features and/or remove potential threats by deploying novel countermeasures.

Achieving elasticity leads to optimal use of SDN and resources. An elastic SDN is simple and easy to deploy, configure, and change. Several works have been proposed in this sense such as distributed controller architectures to dynamically rise or contract the controller pool according to network conditions where the traffic load is reapportioned among controllers [22,51,52]. Other works focus on load-balancing mechanisms [53] or SDN hierarchical methods [54]. For instance, in [22,23], an elastic distributed controller architecture is presented called ElastiCon that utilizes a dynamic adaptation of the controllers' number and their locations in the design of SDN to grow or shrink the controllers' pool depending on the network traffic load. A load-balancing mechanism for the control plane is also considered in ElastiCon as well as in FreeFlow [53]. In [55], a hierarchical SDN/OpenFlow control plane is proposed for mobile cloud computing and Follow me cloud (FMC)-based systems. The elastic control plane is reapportioned into two hierarchical levels: a first level with a global FMC controller and a second level with several local FMC controllers. The local FMC controllers are invoked when the load grows in the global systems.

Table IV illustrates the research papers that deal directly with elasticity issues and challenges and propose solutions and enhancements.

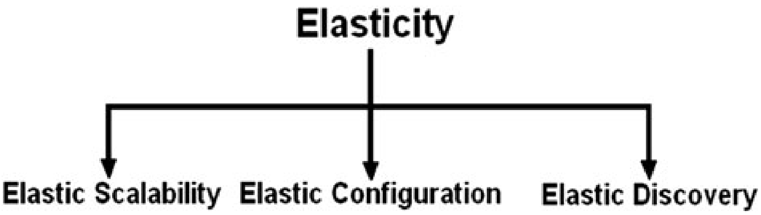

Figure 4. Three dimensions of elasticity.

DOI: 10.1002/sec

Table IV. Elasticity issues, solutions, and challenges.

Reference	Proposal	Main purpose/challenge	Elasticity (technique)
Dixit *et al.* [22,23]	ElastiCon is an elastic distributed controller architecture in which the controller pool dynamically expands or shrinks to automatically balance the load across controllers	Addressing the load imbalance issues to offer elasticity and load-balancing equity	Load-balancing/ distribution
Rajagopalan *et al.* [53]	FreeFlow (split/merge) is a system that allows transparent and balanced elasticity for stateful virtual middleboxes to have the ability to migrate flows dynamically	Providing elasticity by the execution of virtual middleboxes	Load-balancing
Fang *et al.* [54]	Hierarchical SDN is an architectural solution that offers a new way for the forwarding and control planes. Small forwarding tables are used in the network nodes to determine paths	Achieving elasticity and enabling more agile mechanisms for disaster recovery/fast restoration	Hierarchical
Aissioui *et al.* [55]	Elastic distributed SDN controller tailored for mobile cloud computing and FMC-based systems where the SDN/OpenFlow control plane is repartitioned on a two-level hierarchical architecture: a first level with a global FMC controller and a second level with several local FMC controllers	Ensuring elasticity with better control plane management, performance maintenance and network resources preservation	Distribution
Krishnamurthy *et al.* [51]	Pratyaastha is an elastic distributed SDN control plane that permits to efficiently assign state partitions and switches to controller instances	Minimizing intercontroller communication and resource consumption to ensure better performances	Distribution
Mueller *et al.* [56]	A cost-saving approach for on-demand elastic network design and active flow placement in SDN environments based on generic adaptive resource control function to manage topology and control network resources dynamically and on demand	Solving the underlying network design problem using the state-of-the-art integer programming	Mixed-integer programming
Vegad *et al.* [57]	Elasticity analysis of middleboxes application using NFV in SDN environment and SDN elements	Solving elasticity or dynamic scalability issue of NFV (SDN)	(Comparison)

SDN, software-defined networking; NFV, network function virtualisation.

4. DEPENDABILITY, HIGH AVAILABILITY, AND RELIABILITY

4.1. Dependability

We consider dependability to be the ability of a system to provide dependable services in terms of availability, responsiveness, and reliability. Even if a system fails, it can be considered dependable [30], if failures occur with an expected probability. Thus, dependability identifies the trust that can be placed on the service delivered by a system. In this sense, dependability includes concepts in connection with reliability and availability (Figure 5).

The increased complexity in SDN is caused by the interactions between the applications and workloads sharing the physical infrastructure. The prediction of these interactions and adapting the SDN accordingly to provide dependability guarantees in terms of availability and responsiveness is a serious challenge that service providers are faced with. Identifying and deploying the needed resources to meet the service level agreement (SLA) is the most important task to adapt to varying customer workloads and load fluctuations. Also determining the granularity at which resources (e.g., CPU cycles, cluster nodes, additional memory, extended storage capacity, and bandwidth) should be added actively is of paramount importance for long-term respect of SLA.

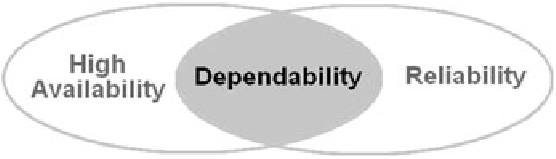

Figure 5. Dependability: high availability and reliability intersection.

Dependability is still an open research problem and represents a strict requirement for the adoption of SDN in practical deployments. However, only few research works can be found in the large body of the literature addressing dependability evaluation of SDN architecture with particular attention to the control plane in different operating conditions.

As dependability is tied to HA and reliability, most of the proposed works deal with the two primitives and propose various solutions, but we can only find some few straightforward researches that shed light on dependability issues and evaluate the impact of load fluctuation on SDN dependability as demonstrated in Table V.

Table V. Dependability issues, solutions, and challenges.

Reference	Proposal	Main purpose/challenge
Longo *et al.* [58]	A stochastic model to evaluate reliability and availability (dependability) of SDN architectures, and represent SDN controller whose components are organized in a hierarchical topology. This model is based on an algorithm that represents realistic behaviors and changing operating conditions of the network because of workloads	Presenting and evaluating a dependability model of SDN to pinpoint the key aspects of reliable and available SDN architecture. In particular, the evaluation is mainly based on a parametric controller model, instantiated and populated with different parameters and measures to evaluate the control plane dependability and reliability
Wu *et al.* [59]	A probabilistic model to analyze dependability of SDN with the PRISM tool. The results of the designed system model are verified and visualized using PRISM. SDN system is modeled using Markov chain model	Improving the dependability (reliability) of SDN by implementing multicontroller architecture
Kreutz *et al.* [60]	Discussing the important aspects of a secure and dependable SDN architecture as well as threats/ vulnerabilities. A brief description of the mechanisms required to build a secure and dependable SDN control platform is also introduced	Dependability is of paramount importance when designing SDN

4.2. Service level agreement

Although SLAs [31] commonly include details of the agreed aspects of the service such as scope, quality, performance, time delivery, and speed of memory and CPUs, there is no mechanism that could prevent the failure of one or more elements in the SLA contract. In fact, some system or network misbehaviors and glitches are unpredictable owing to bugs, fake process, and/or malwares [61] running in the background, which might cause high CPU and memory usage. Consequently, services might be unreachable for end customers. Another issue that may emerge is purely related to hardware availability [62]. Major cloud providers have experienced complete blackouts because of minor misconfigurations or dysfunctionings as reported in [20,21].

The SLA typically dictates advanced technical specifications in terms of HA such as throughput, bandwidth, transfer rates, jitter and latency, and HA metrics: mean time between failures (MTBF), mean time to recover (MTTR), mean time to repair (MTTRr), mean time to failure (MTTF), and mean time to detect (MTTD). MTBF is the "up-time" between inherent failures of a repairable system during operation. MTTR is the average time that a device/ network will take to recover from any failure. MTTRr is the average time to repair the failed device after identifying the root cause of the malfunctioning. MTTF is the mean time to failure starting from the moment the device recovers. MTTD is an expected average time to detect a failed network component (Figure 6).

The availability of SDN can be calculated using the following formula:

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTBF}} = \frac{\text{MTBF} - \text{MTTD} - \text{MTTR}}{\text{MTBF}}$$

$$\text{where} \begin{cases} \text{MTTR} = \text{MTTD} + \text{MTTRr} \\ \text{MTBF} = \text{MTTR} + \text{MTTF} \end{cases}$$

Percentages of availability are sometimes referred to by the class of nines (Table VI). For example, three nines means nearly 9 h of unavailability per year.

4.3. High availability: challenges

High availability is generally one of the most challenging goals in an operational cloud. In the conventional network, we can find several HA mechanisms (e.g., link aggregation, multipath routing, system/network redundancy and backup, state synchronization, failure detection and handling) and protocols (e.g., link aggregation control protocol [63], EtherChannel [64], equal-cost multipath routing [65], virtual router redundancy protocol [66], host standby router protocol [67], resilient packet ring [68], nonstop routing [69], nonstop forwarding [70], stateful switch-over [71], Ethernet automatic protection switching [72], Ethernet ring protection switching [73], and fast rerouting [74]).

Like every component of the cloud, SDN needs to be run continuously to meet the HA objectives that aim at

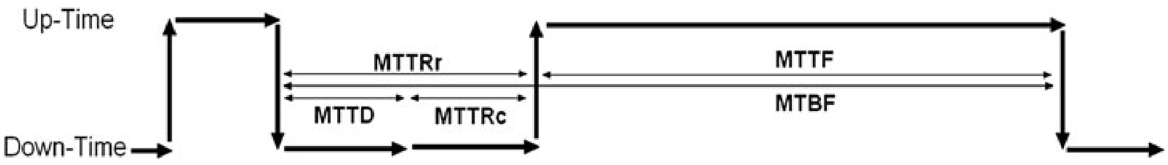

Figure 6. Software-defined networking availability timeline.

DOI: 10.1002/sec

Table VI. Percentage of availability.

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90 (“1 nine”)	36.5 d	72 h	16.8 h	2.4 h
99 (“2 nines”)	3.65 d	7.20 h	1.68 h	14.4 min
99.9 (“3 nines”)	8.76 h	43.8 min	10.1 min	1.44 min
99.99 (“4 nines”)	52.56 min	4.38 min	1.01 min	8.66 s
99.999 (“5 nines”)	5.26 min	25.9 s	6.05 s	864.3 ms
99.9999 (“6 nines”)	31.5 s	2.59 s	604.8 ms	86.4 ms
99.99999 (“7 nines”)	3.15 s	262.97 ms	60.48 ms	8.64 ms
99.999999 (“8 nines”)	315.569 ms	26.297 ms	6.048 ms	0.864 ms
99.9999999 (“9 nines”)	31.5569 ms	2.6297 ms	0.6048 ms	0.0864 ms

providing nonstop physical and virtual resources (e.g., applications, platforms, databases, and computational servers). However, the separation between the data and control planes, as well as centralizing the controllability in one entity, has brought new challenges. Most of the aforementioned mechanisms and protocols can be applied to the cloud infrastructure, but unfortunately, they do not support specific SDN feature requirements such as correlation of failures between the control plane and the data plane, HA interconnection, and synchronization between the SDN controllers. Obviously, dealing with the brain of the SDN is more crucial than with a decentralized network where responsibilities are shared and partitioned among every component. Controllers are critical as long as HA is concerned because decisions and orders are in their hands, which makes them a single point of failure. So far, few HA related works for data and control plane have been proposed as illustrated in Table VII.

Table VII. HA issues, solutions, and challenges.

Reference	Data plane	Control plane	Proposal	Main purpose
Williams *et al.* [83]	✓		RuleBricks provides HA in existing OpenFlow policies	Embedding HA policies into OpenFlow’s forwarding rules
Desai *et al.* [84]	✓		An algorithm: checking connectivity among neighboring switches for fast failure detection by using the OpenFlow switch’s link signal	Fast failure detection in SDN
Kempf *et al.* [46]	✓		Extending the OpenFlow protocol to support a monitoring function on OpenFlow switches	Achieve fault management and recovery in the data plane
Kim *et al.* [75]	✓		CORONET (controller-based robust network): SDN fault-tolerant architecture	Recovering from multiple link failures in the data plane
Borokhovich *et al.* [76]	✓		Local–fast failover algorithms	Guaranteeing quick reestablishment of connectivity in the data plane
Heller *et al.* [77]	✓		Placing OpenFlow switch to the closest controller in the network	Optimizing the number of controllers and their location in the network
Tootoonchian *et al.* [17]		✓	HyperFlow: a distributed event-based control plane for OpenFlow	Redundancy and minimizing the control plane response time to data plane requests Resilience to network component failures
Koponen *et al.* [41],		✓	Onix: a distributed control platform that provides a general-purpose API for control plane implementations	Turning networking problems into resolvable distributing systems problems
Berde *et al.* [85]		✓	ONOS (open network operating system): a distributed SDN control platform	Providing HA and scalability through a distributed SDN control

(*Continues*)

Table VII. (Continued)

Reference	Data plane	Control plane	Proposal	Main purpose
Cai *et al.* [100]		✓	Maestro: a controller for scalable OpenFlow network control	Implementing parallelism to improve the capacity of the controller.
Sharma *et al.* [78]		✓	Fast failure recovery mechanism in OpenFlow networks	Fast restoration mechanism in OpenFlow
Park *et al.* [79]	Link between data plane and control planes	Control path HA algorithms between the control and data plane Virtualized controller cluster	Fast failure detection and recovery to ensure high availability of SDN	

HA, high availability; SDN, software-defined networking.

Furthermore, redundancy of controllers is a fundamental pillar in the HA concept. There are some methods [80] to achieve this in real time in order to ensure HA and reliability. Designing a redundant architecture is one of the main challenging operations in a highly available SDN. Additionally, latency [81,82] is another major issue facing the network providers. The latency does not only affect the traffic but also the failover times, and thus, this comes at the cost of HA. There are several causes of low latencies, among which are low power RAM/CPU in network component, longer network update/upgrade time, congestion, and insufficient bandwidth. Basically, latency in SDN can be divided into three categories:

(1) Data plane latencies: Switches and routers or any other network components in the data plane have their specific characteristics that determine their capacity in terms of memory, CPU, and so on. For instance, in an OpenFlow (OF) switch there is an inbound and an outbound latency from the controller perspective:
   (a) *Inbound latency:* Represents the time between receiving a packet from the data plane and sending it to the controller. First, the switch relies on the forwarding engine to send packet for a lookup in the rule table. Here, two cases may emerge: In case of (i) reactive OpenFlow operation mode, the OpenFlow agent performs a lookup in the flow tables. If there is no match, the switch automatically creates an OpenFlow "packet-in" packet and sends it to the controller for orders. (ii) In proactive mode, flow tables are already predefined and thus no need to consult the controller for instructions. This could be advantageous in eliminating the latencies resulting in contacting a controller.
   (b) *Outbound latency:* The switch receives the flow from the controller and then parses OpenFlow messages that hold new rules to be applied to the OpenFlow switch's flow table. The whole update process could take a long time.

There is another mode of operation called Hybrid mode when OpenFlow is used. In this mode, the combination of proactive and reactive mode has been made to allow more flexibility and maintain low latencies.

(2) *Control plane latencies*: In the control plane, the latencies associated with an SDN controller are Input/Output (I/O) performance, processing, and flow setup time. These performance metrics heavily influence when additional SDN controllers are deployed. If the OF switches fire out too much flows, the controller would take longer time to respond, and thus, the latency is increased.
(3) *Interconnection latencies*: These low latencies are measurable and depend mainly on the physical medium capacity (i.e., bandwidth and throughput). Generally, interconnection latencies are negligible.

In [83], the authors presented RuleBricks that tackles HA issues in the data plane for servers. In [84] and [46], the authors proposed fast failure detection and fast recovery mechanisms. Other works focus on management in the control plane such as those in [41] and [85]. An extended list of related works with further details is presented in Table VII.

4.4. Reliability

Reliability is another important aspect of SDN where the delivery should be notified in case of failures. The service providers deploy reliable applications and services that notify the end user of an eventual delivery failure. Reliability specifications indicate the guarantees that the delivery of data must concern only the intended recipient through reliable SDN architecture and reliable protocols (Table VIII).

Improving reliability is of paramount importance because network failures could easily cause disconnections between the control and forwarding plane. A reliable SDN should work continuously under an augmented workload without dropping messages from the control plane or the data plane.

Table VIII. Reliability issues, solutions, and challenges.

Reference	Method/description	Challenge/main purpose
Shalimov *et al.* [19]	Conducting efficiency tests with HCPROBE framework to show whether the SDN/OpenFlow controllers are reliable in production environment	Reliability analysis reveals that NOX, POX, Beacon, Floodlight, MuL, Maestro, and Ryu are capable of dealing with average workloads
Hu *et al.* [86]	Placing controllers in SDN to maximize the reliability of SDN control plane and considering it as a novel reliability metric called expected percentage of control path loss and analyzes the dependent control path failures in SDN	Showing that the placement of controller is not the only key to reliability. This latter depends on properly choosing the number of controllers
Yao and Bi [87]	The cascading failures of controllers can affect the reliability of SDN. In fact, the load of the controllers that carries the load of the failed controller can exceed their capacity, and then cascading failures of controllers will likely occur	Disclosing failure threat to reliability of SDN networks and proposing a model of these failures to prevent them
Ros *et al.* [88]	Determining the fault-tolerant controller placement using heuristic mechanism that executes placement algorithm in order to achieve at least "5 nines" reliability in the southbound interface between controllers and nodes	Deploying fault-tolerant SDNs by achieving 99.999% of southbound reliability between controllers and nodes (i.e., switches, routers, and middleboxes)
Xiao *et al.* [89]	An approach based on partitioning a large network into several small SDN domains by using the spectral clustering placement algorithm to solve SDN controller placement problem for WAN	Maximizing the reliability of controller and minimizing the latency of WAN using spectral clustering placement algorithm
Capone *et al.* [90]	A failure management framework for SDN that uses OpenState (an OpenFlow extension) capabilities to allow fast path restoration. The framework can handle link and node failures. When a failure is detected on a link, packets are recovered through a detour path	Enabling reliability through a reliable and scalable mechanism named OpenState to provide protection against a link or node failure
Pfeiffenberger *et al.* [90]	Exploiting features of OpenFlow to provide one-link fault tolerance with negligible packet loss. The used approach allows low recovery times from link failures and continuity of fault tolerance	Evaluating the use of SDN for reliable communication

SDN, software-defined networking.

The reliability of the control plane is very critical for SDN because it is the brain of the production network. If the control plane fails, the SDN networks will systematically lose packet forwarding and running processes. SDN reliability [92] is hard to analyze because of its characteristics of software and hardware components and complicated interactions among them.

There are several types of failures that may influence the reliability of SDN such as overflow, timeout, unreachable resources, software failure, database failure, hardware failure, and link failure. These types of failures might be dependent and should be considered when designing reliable SDN networks. For example, network failures may disable invocation among software programs to retrieve the necessary informations.

Service providers should isolate faulty elements in case of failures and minimize their impact on the quality of experience [93] in order to respect SLA and satisfy the customers.

5. PERFORMANCE

Software-defined networking offers flexible and programmable functionalities as well as centralized control, better user experience, and a decrease in network systems and equipment costs. However, the offered capabilities and features of SDN infrastructure influence the whole network performance (e.g., latency, jitter, throughput, and error rate). In fact, SDN may pay performance penalties due to workload of the network traffic that the OpenFlow-based SDN [26] systems handle. The control plane is the main bottleneck of the SDN architecture where performance is a critical issue especially in large-scale network implementations because a heavy request load on an SDN controller might result in longer delays.

Substantially, facing the challenges of SDN performance is following a mainstream pattern by deploying new solutions for classical issues. Many existing mechanisms can still be used in order to properly manage SDN performance such as quality of service [94], load balancing [95], end-to-end congestion control [96], and data traffic scheduling [97].

The performance analysis of network controllers is generally carried out through benchmarking. Benchmarking is a commonly used method for analyzing the performance of the controllers as well as identifying performance bottlenecks in order to take a decision of increasing power processing capabilities of a controller. In several works, the benchmarking procedure has been performed on existing SDN controllers [43,98,99]. An improved version of NOX has been developed for better performance, which is known as NOX-MT. Various [99] controllers have been tested for performance including Floodlight, Beacon, NOX-MT, and Maestro. The authors concluded that the performance of OpenFlow-based controllers varies, depending on architectural features like packet batching, switch partitioning, and multicore support. They have noticed that the controllers that implement static switch partitioning and static batching have high throughput performance. In contrast, controllers with adaptive batching technique have reasonable latency performances. Benchmarking on NOX, NOX-MT, Maestro, and Beacon has shown performance advantages of NOX-MT over the others in terms of minimum and maximum response time, as well as maximum throughput. The NOX-MT controller exploits multithreading based on the single thread C++ implementation of the network operating system (NOX) [43]. The NOX-MT uses optimization techniques such as I/O batching to improve baseline performance. These optimizations allow NOX-MT to outperform NOX by a factor of 33 on a server with two quad-core 2 GHz processors. Maestro [18,100] is a Java-based controller implementation. It utilizes both parallelism and throughput optimization technique (e.g., input and output batching and core and thread binding). It was revealed, in benchmarking, that the Maestro design can guarantee good performances and near linear performance scalability on multicore processors. In [101], the authors have used the standard Cbench [27] tool to conduct performance tests in terms of latency and throughput of both OpenDaylight and Floodlight controllers. They have found that benchmarking results of OpenDaylight with Cbench have not been very successful and claimed that the controller experienced several memory issues like memory leakage. The Cbench tool tests controller performance by generating requests for packet forwarding rules and watching responses from the controller. Cbench offers aggregated statistics of controller throughput and response time for all the switching devices. Aggregated statistics may not be sufficient enough to explore detailed controller behavior. Consequently, another tool named OFCBenchmark [28] is considered to enable fine-grained statistics. OFCBenchmark provides statistics of response rate, response time, and number of unanswered packets for individual switching devices.

Benchmarks have been also conducted for the data plane components. In [102], the authors compared OpenFlow switching, layer-2 Ethernet switching, and layer-3 IP routing performance and showed higher forwarding performance and better fairness of Linux software SDN switching. As performance indicators, they have used forwarding throughput and packet latency forwarding throughput and packet latency under overload.

In [103] and [104], mechanisms have been proposed to increase the performance of OpenFlow using an architectural model to improve the lookup performance of OpenFlow switching. In [103], packet switching throughput increases to 25% compared with the throughput of regular software-based OpenFlow switching. However, in [104], the authors have used network processor-based acceleration cards to perform OpenFlow switching. The results have shown 20% reduction on packet delay compared with classical models.

A framework called OpenFLow Operations Per Second (OFLOPS) has been introduced to facilitate performance benchmarks. OFLOPS supports multiple packet generation and capturing mechanisms [105]. OFLOPS allows measurement of performance metrics for control plane operations such as rule insertion delay and traffic statistic query delay as well as detailed performance measurement data plane operations of SDN switching devices.

In the literature, three main methods have been proposed to enhance the performance of SDN: The first one is by handling request in the data plane on behalf of the controller [16,39], the second method is by structuring the switching devices [42], and the third is by implementing power processing mechanisms such as multiple-core processing and mutithreading [43]. When it comes to SDN, the tradeoffs between high performance and programmability/flexibility should be carefully considered (Figure 7 and Table IX).

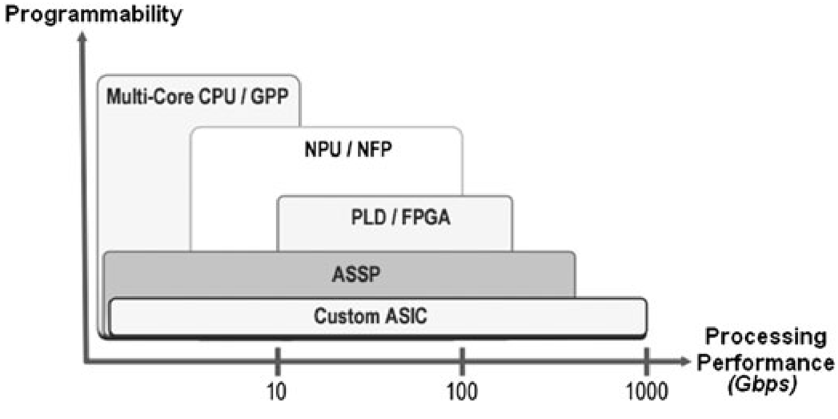

Figure 7. Performance and programmability.

Table IX. Performance issues, solutions, and challenges.

Reference	Method/description	Challenge/main purpose
Yu *et al.* [16]	Request handling in the data plane by distributing rules across "authority switches"	Ensuring low latencies by eliminating the need to ask the controller for rules
Curtis *et al.* [39]	DevoFlow installs a given set of possible packet forwarding rules in switching devices	Offering equal-cost multipath routing in case of port failures
Yeganeh and Ganjali [42]	Kandoo is a layered architecture: The bottom layer is a cluster of controllers with limited network view that handles frequent events. The top layer is a logically centralized controller that has clear view and only handles rare events. Heavy communication only occurs at the bottom layer	Improving the overall control layer performance and scalability by reorganizing the structure of switches
Tootoonchian *et al.* [43]	Enhancement of network operating system (NOX) to support multithreading	Increasing performances in terms of response times as well as throughput
Khattak *et al.* [101]	Performing benchmarking on OpenDaylight SDN controller and comparing latency and throughput results with Floodlight's benchmarks	Carrying out a performance analysis and an evaluation of OpenDaylight SDN controller
Zhou *et al.* [106]	Using a caching mechanism for data network applications with SDN to speed up the service of northbound REST API	Increasing the performance and maintaining flexibility of northbound REST API
Egilmez *et al.* [107]	OpenQoS is an approach for video streaming over OpenFlow networks with dynamic QoS routing to fulfill end-to-end QoS support, which is possible with OpenFlow's centralized control capabilities over the network	Guaranteeing seamless video delivery through prioritization in the control plane
Xiong *et al.* [108]	Investigation of SDN's performance management of analytical queries in distributed data stores in a shared networking environment and analyzing the priority of network traffic in order to make network bandwidth reservations	Increasing performance management of analytical queries in distributed data stores through bandwidth reservations using queuing theory
Wendong *et al.* [109]	AQSDN: in AQSDN architecture, QoS features can be configured autonomically in an OpenFlow switch by extending the OpenFlow/OF Config protocols. A QoS model named packet context-aware QoS is suggested to improve the network QoS	Introducing a packet context-aware QoS model to provide self configuration and high QoS in AQSDN architecture
Machado *et al.* [110]	Policy refinement approach for QoS management: a method that is capable of identifying QoS requirements and the resources that need to be configured in accordance with the SLAs based on uses of policy-based management	Aiming at removing manual workload of configuring network elements and implementing reactive dynamic actions to reconfigure the SDN components and meet the SLA requirements
Jarschel *et al.* [26]	Analyzing the processing time, the forwarding speed, and the blocking in the forwarding queue to the controller	Presenting a model that helps in showing the importance of the controller performance
Jarschel *et al.* [28]	OFCBenchmark tool helps in conducting flexible benchmarks to analyze the SDN controller performance	Achieving a flexible OpenFlow controller benchmark
OF group [111]	Performance tests of NOX Beacon and Maestro controllers	Providing a performance comparison of SDN controllers: NOX Beacon and Maestro
Liu *et al.* [112]	Performance analysis of GMPLS-based, PCE/ GMPLS-based, and OpenFlow-based control planes for a translucent wavelength switched optical network	Analyzing the integration of the PCE with GMPLS

AQSDN, autonomic QoS management mechanism in software-defined network; SDN, software-defined networking; QoS, quality of service; SLA, service level agreement; PCE, path computation element; API, application programming interface; GMPLS, Generalized multiprotocol label switching.

6. RESILIENCE

Achieving resilient communication is a top purpose of networking. As such, SDNs are expected to yield the same levels of reliability as a legacy and any new alternative technology. SDN architecture is resilient when it has the ability to operate under load, failures, and attacks. Generally, resilience is based on both reliability and security concepts. As far as SDN is concerned, resilience is the ability of one or more components of the SDN architecture to, either in the control plane or data plane, recover quickly and continue operating even when there has been an equipment failure, power outage, or other disruption. SDN resilience should be planned as a part of SLA to ensure continuous computing services.

As the SDN network can be prone to failures, the development of the OpenFlow protocol version 1.2 has included the capability of applying master–slave configuration to deal with faulty OpenFlow controllers and increase resilience. This indicates that a master controller has control over all switches and on the master failure state; a backup controller (slave) can take the lead through a timely detection of failure and fast failover process.

To build a resilient network, the network topology must consider different challenges [29]. For instance, it must include redundant paths between network nodes and ensure an identical synchronized network state between master and slave controllers.

Additionally, the inclusion of new modules and components into the network should abide by the requirements without degrading the network performance and causing failures.

Several works have been proposed to increase network resilience (Table X). The proposed research papers mainly deal with replication [113], overlay networks [96,114,115], multihoming [115,116] and multipath networking [89–91].

Resilience can be modeled as an unforeseeable cycle that starts from an SDN normal state, where every network node behaves as intended and delivers an acceptable service and communicates without any disruption under normal operation $S_{0,}$ to unacceptable service delivery under severely degraded service S_c. Anomalies are detected early to prevent failures or at least enable a partially degraded network to recover afterwards. However, escaping from severely degraded zone S_c to initial state S_0 requires a successful remediation and then a fast recovery. Figure 8 illustrates a multilevel two-dimensional state space that is characterized by two axes representing the operational state N (horizontal) and service parameters P (vertical). This model is commonly used to identify potential network threats and failures so as to anticipate and react in real time to ensure network resilience.

Table X. Resilience issues, solutions, and challenges.

Reference	Proposal	Main purpose/challenge	Resilience (technique)
Fonseca *et al.* [113]	Active and passive replication: In passive mode, the client connects with one controller that handles the requests and informs the rest of controllers. In active mode, the client communicates with multiple controllers	As SDN relies on the control plane, this poses the issue of a single point of failure. With replication techniques, SDN can be more resilient to failures	Replication
Rohrer *et al.* [117]	Path diversification to select multiple path metrics for evaluating path, node pair, and graph diversity. The path diversification mechanism is targeted at the end-to-end layer but can be applied at any level for which a path discovery service is available (e.g., intrarealm routing or interrealm routing)	Improving flow robustness in the presence of link and node failures. Path diversification provides a substantial performance improvement over conventional single-path mechanisms	Multipath
Zhang *et al.* [114]	ROMCA allows combining centralized topology construction control and distributed dynamic mapping of paths onto the overlay topology according to network conditions	Mitigating the shortcomings of the network infrastructure and providing resilience with low recovery times in response to network failures	Overlay-based
Akella *et al.* [116]	Multihoming where data centers or enterprise networks are connected to multiple providers could increase network resiliency if the right ISPs are chosen	Improving network performance and resilience through multihoming	Multihoming
Han *et al.* [115]	Using source-based single-hop overlay routing combined with a topology-aware node deployment and increasing the usage of multihoming environment	Designing a topology-aware network overlay using multihoming in order to enhance the resilience properties	Overlay-based/ multihoming
Li *et al.* [118]	SmartTunnel is a logical end-to-end tunnel between two end points that combines several physical network paths. These paths forming the tunnel are chosen to be topologically diverse	Ensuring resilience by strategically allocating traffic onto multiple paths and performing forward error correction coding	Multipath
Chiu *et al.* [119]	CORONET is an SDN fault-tolerant system that recovers from multiple link failures in the data plane. CORONET uses information about the network state coming from physical network routers, in particular, optical network routers	Using multipath in CORONET to provide a complete fault-tolerant system that recovers from failures occurring in other fault domains in SDN-based networks	Multipath

CORONET, controller-based robust network; SDN, software-defined networking.

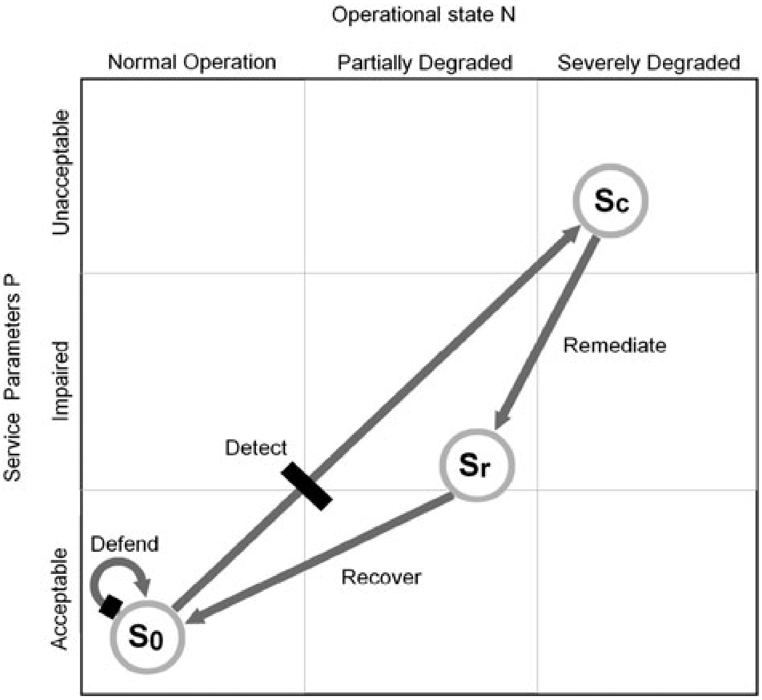

Figure 8. Multilevel two-dimensional state space.

7. SDN SECURITY

7.1. Software-defined networking: vulnerabilities

While SDN provides a new network design enabling a wide range of network applications and services, security concerns have become an important pillar, as security is not yet a built-in feature in the SDN architecture. Being based on existing technologies (e.g., routers, switches, servers, applications, etc.), it inherits systematically its related security issues ranging from physical to application layer of the Open Systems Interconnection model. However, SDN has brought new network elements (e.g., controllers, virtualized components, etc) and various applications into the core network with a new splitting design.

Decoupling the data plane from the control plane would eliminate some existing minor threats in traditional network, but it has created new loopholes specific to SDN.

As SDN mainly relies on various software in the application layer to interact with the underlying infrastructure, this has surely major drawbacks because code vulnerabilities can have a serious security impact. Additionally, by making the controller as the central location for management and control, this would result in creating a point of failure for the whole infrastructure. Controllers could be threatened from different sides (i.e., interfaces). The southbound and the northbound interfaces are respectively the entrance doors to the control, which themselves might be an appealing target for attackers. The data plane is also a potential target for attackers. Because of the exchanges between the control and data planes, an attacker could disrupt the data flow by flooding the switches or tampering the communications. The OpenFlow protocol responsible for establishing communications between the two planes can also be vulnerable to some attacks [120].

The SDN vulnerabilities [24,25,120–126] can be repartitioned into five major axes (Figure 9):

1) *Application layer*: These are software-related vulnerabilities that might be exploited through code injection. Some works have been proposed to tackle common issues of the application layer. Such issues include the challenge of detecting and reconciling potentially conflicting flow rules from OF apps (FortNOX [127]) and dealing with network applications' failures leading to loss of network control (ROSEMARY [128]), and enabling network resilience to SDN application failures and faults (LegoSDN [129]).
2) *Control layer*: This includes vulnerabilities associated with the controller. Major works have been proposed in the literature to identify and mitigate/reduce vulnerabilities of the controller. Othman and Okamura [130] present a signature algorithm to securely transmit flow installation requests from the controller to the network devices. The system assumes using secure communication with TLS so as to avoid the control channel from being tampered. In [131], the authors proposed an

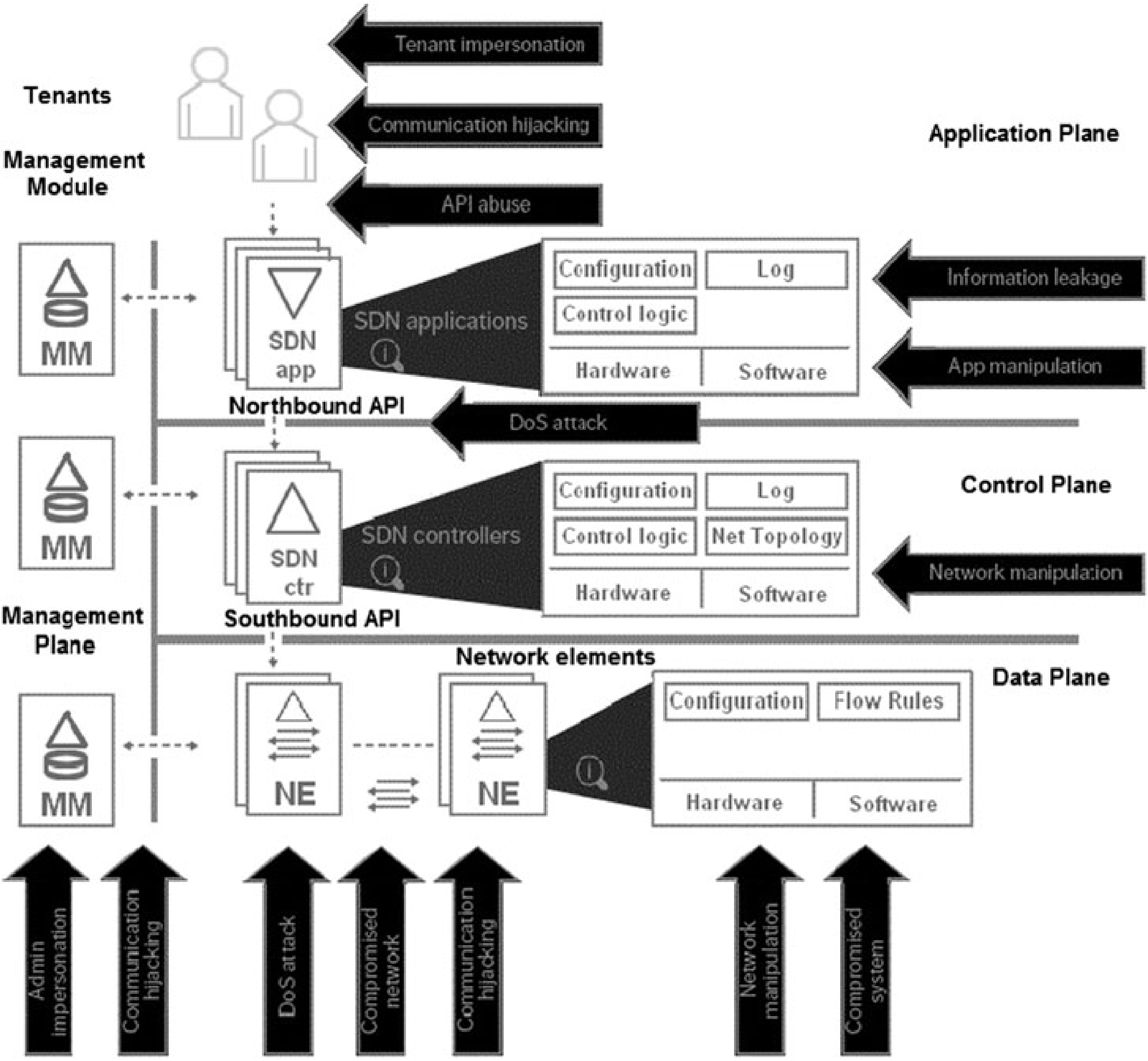

Figure 9. Various threats/attacks against SDN.

algorithm that offers a secure and resilient SDN architecture through the implementation of several fault-tolerant controllers. A hierarchical system of controllers is identified in [132] to reduce the impact of a malicious application that can lead to point of failures in the control plane. Other works such as formal methods [133–136] have been proposed to verify and prove the safety and correctness of the SDN control plane and resolve issues related to security policies.

3) *APIs*: The control plane exposes two types of APIs: vertical (i.e., southbound and northbound) and horizontal (i.e., eastbound, westbound, and management interface). The issue of exposing interfaces is discussed in [137]. The authors proposed PermOF that applies minimum privilege on the applications to enforce permissions at the API entry. Similar works have been introduced in [138] and [139], where the authors have mainly pinpointed the lack of security in the communication processes between OpenFlow applications and the control plane. As a solution, the Security-Enhanced Floodlight is proposed that offers an authenticated northbound API. Another system named Frenetic [140] also focuses on the northbound API to resolve policy conflicts.
4) *Protocols*: In OpenFlow-based SDN architecture, the OpenFlow protocol can be exposed to various threats according to vulnerability assessments [120], among which is the TLS security layer that has been considered as optional in the specification of the OpenFlow protocol. A definition of security requirements and a detailed analysis of the OpenFlow protocol are presented in [25] and [8].
5) *Infrastructure layer*: Vulnerabilities related to data plane components (e.g., switchs, and routers) could

engender different attack vectors such as fraudulent rule insertion, rule modifications, and flooding [120]. FlowChecker [141] has been introduced to solve issues of misconfigurations in OpenFlow switches causing variance in flow rules. Anteater [142] is a similar work that introduces a static analysis approach to diagnose network configuration problems. OFHIP [143] is proposed to provide secure mobility with OpenFlow to avoid disrupted flow processing, active TLS session teardown, and mutual authentication issues.

7.2. Software-defined networking: threats and attacks

Vulnerabilities in the management, application control, and data planes entail different threats for the SDN model and could be classified into four categories depending on the motivations:

(1) *Internal*: Internal threats come from the inside of SDN, generally from authorized individuals who can have full access to the SDN network (e.g., managing applications, modifying security policies, programming scripts for SDN, etc). According improperly high privileges can lead to serious malicious actions [127,137,144] into the network either being intentional or accidental. Although intentional malicious activities are rare, they are the most dangerous threats for a productive network. Accidental undesired network manipulations are also non-negligible, which can cause undesired traffic or network loops.

(2) *External*: External threats are mainly initiated from the outside of the corporate network and aimed at altering the SDN network by relying on denial of service (DoS) attacks (Table XVI) or trying to have unauthorized access (Table XV) by relying on advanced techniques such as malwares [61], social engineering, man in the middle, and spoofing (Table XII).

(3) *Unstructured*: Generally, this type of threat comes from script kiddies, bugs/misconfigurations, or conflicting rules [141,145] in the SDN components (e.g., applications, security policies, conflicting rule tables in the switches, etc.).

(4) *Structured*: This threat can entail very sophisticated attacks leading to serious damages in the SDN network. In fact, this threat comes from high-level technically skilled attackers who can penetrate into networks by varying their advanced techniques (e.g., spoofing, code exploits, social engineering, malwares, etc.). Skilled hackers can join their forces and lead an organized crime against organizations and network providers.

The attackers can lead one or more attacks depending on their motivations and skills. Generally, attacks against SDN can be classified into two types: passive and active

Table XI. Anomalies and scanning attacks.

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Mehdi *et al.* [146]	Anomaly detection algorithms that use OpenFlow flow information for traffic anomaly detection including the detection of scanning worms	Exploiting programmability of SDN to provide accurate anomaly detection capabilities in home networking	IN/EX ST UST	AC/PS	CT/SB/AP
Hand *et al.* [147]	Exploiting the programmability of SDN infrastructure to provide active security by sensing and countering threats	Network control in order to detect intrusions, memory content capture, and scanning	IN/EX ST UST	AC/PS	CT/SB/AP/DT
Jafarian *et al.* [148]	OpenFlow Random Host Mutation where hosts' identities should be continuously and randomly changed by mutating IP addresses	OpenFlow Random Host Mutation deceives hosts' scanning attacks and IP address discovery	IN/EX ST UST	AC/PS	CT/SB DT
Kampanakis *et al.* [149]	SDN-based moving target defense network protection enables increasing the attacker's overhead and obfuscation to confuse the attacker	Providing protection against reconnaissance, fingerprinting, and service discovery	IN/EX ST UST	PS	CT/SB/AP/DT
Giotis *et al.* [150]	Solution architecture combines OpenFlow and sFlow where flow statistics are gathered and analyzed to identify anomalies that are neutralized by an anomaly mitigation module	Enabling anomaly detection and mitigation against network attacks such as port scanning attacks, worm propagation, and overloads due to potential DoS attacks	IN/EX ST UST	AC/PS	CT/SB/AP/DT

Type of threat–EX, External; IN, Internal; ST, Structured; UST, Unstructured. Type of attack–AC, Active; PS, Passive. SDN layer/API–CT, Control plane; DT, Data plane; SB, southbound; NB, northbound; OF, OpenFlow; AP, application plane; SDN, software-defined networking.

attacks. In a typical scenario, a sophisticated attack combines the two modes of attacks. In the first mode, the attacker is only interested in monitoring, gathering information, eavesdropping message contents, and analyzing the traffic in order to identify IP addresses, open ports, and active services and communications by using scanning tools. Afterwards, on the basis of the collected data about the network and the identified potential holes, he or she can proceed to an active attack so as to gain illegitimate access for spying and/or exhaust a resource or service in the SDN infrastructure, which is known as a DoS attack (Tables XI–XVI).

Several countermeasures have been proposed in the literature to eliminate or mitigate SDN threats in the

Table XII. IP and ARP spoofing.

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Matias *et al.* [151]	Address resolution mapping is an embedded module in the controller that verifies MAC addresses from authorized users or hosts by tracking them	Mitigating ARP poisoning attacks that could take place between the controller and OF switches	IN/EX ST	AC/ PS	CT/ SB/ DT
Yao *et al.* [152]	Solving the address resolution problem with the help of OpenFlow VAVE. VAVE is a module that examines the records in the flow table	Enabling protection feature against IP spoofing, on a NOX-based OpenFlow controller, which could lead to DoS attack	IN/EX ST	AC/ PS	CT/ SB/ DT
Yao *et al.* [153]	Software-defined filtering architecture is a method based on VAVE. Software-defined filtering architecture allows adding filtering rules based on spoofing occurrences	Enhancing VAVE to counter IP spoofing by applying new filtering rules	IN/EX ST	AC/ PS	CT/ SB/ DT
Feng *et al.* [154]	Extending the solution of VAVE using OpenRouter where every router has a global network view of address assignment	Adding the support of IP spoofing detecting through OpenFlow routers	IN/EX ST	AC/ PS	CT/ SB/ DT

SDN, software-defined networking; VAVE, virtual source address validation edge; API, application programming interface; ARP, Address Resolution Protocol.

Table XIII. Tampering and information disclosure.

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Meyer and Schwenk [155]	Overview on attacks and countermeasures on the TLS handshake protocol, the TLS record and application data protocols and the PKI infrastructure of TLS	Identifying current SSL/TLS security concerns in OpenFlow. SSL/TLS is an optional security feature	IN/EX ST	PS	CT/ SB/ OF
Benton *et al.* [120]	Overview of man in the middle attacks that are currently seen to be a significant possible threat in the current OpenFlow architecture	Discussing man in the middle attack, which is an information disclosure attack that targets information in transit	IN/EX ST	PS	CT/ SB/ OF
Schehlmann and Baier [179]	COFFEE is a framework that enables detection and mitigation of botnets. The detection is based on NetFlow to filter, detect, and alter SDN components	Countering botnets that can lead to data modification or disclosure	IN/EX ST	PS	AP/ CT/ SB/ OF
Kloti *et al.* [25]	Security analysis and modeling methodology for the OpenFlow protocol and network using STRIDE [156]	Discussing vulnerabilities such as information disclosure and denial of service	IN/EX ST	PS	AP/ CT/ SB/ OF

PKI, Public Key Infrastructure; SSL, Secure Socket Layer.

Table XIV. Privilege escalation.

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Porras *et al*. [127]	Fort-Nox is a flow-based authorization system that provides various security components such as security audit trail for flow rule commands, rules' conflicts, and resolution outcomes by exploiting information like application ID, privilege level, and flow time	Fort-Nox helps in detecting and handling conflicts of flow rules from OF apps and providing security enforcement kernel for prioritizing flow rules with role-based authorization	IN/EX ST	AC/ PS	CT/ SBNB/ OF
Wen *et al*. [137]	PermOF is an access control system that relies on a set of permissions and an isolation technique to deal with permissions at the API entry by applying privilege restrictions to applications	Protecting SDN from privilege escalation and unauthorized access on the control plane	IN/EX ST	PS	AP/NB
Hayward *et al*. [144]	Operation-Checkpoint is a permission system that defines a set of permissions against which applications must be checked before being authorized to execution	Limiting the exposure of the control plane to application plane in order to avoid SDN application layer attacks	IN/EX ST UST	AC/ PS	AP/ CT/NB

Table XV. SDN intrusions and unauthorized access.

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Casado *et al*. [157]	SANE system provides a single protection layer where controller includes access control rules instead of having them in firewalls as in traditional networks	Logically centralizing routing and access control decisions to allow or deny access. SANE offers more visibility on security policies	IN/EX ST UST	AC/ PS	CT/ SBDT
Jia and Wang [158]	SDN-based firewalls for P2P networks that are provided as an API for SDN. Due to the nature of P2P networks, SDN requires specific security mechanism to prevent intrusions	Network demand is frequently changing and P2P firewall solution should be integrated	IN/EX ST UST	AC/ PS	CT/ SBDT/ AP
Katta *et al*. [159]	Flog implements stateful firewalls in order to inspect connection states from insiders by analyzing already established communications	Implementation of stateful firewalls to detect possible malicious code	IN/EX ST UST	AC/ PS	AP/ NBDT/ CT
Hayward *et al*. [144]	Operation-Checkpoint is a permission system that defines a set of permissions against which applications must be checked before being authorized to execution	Limiting the exposure of the control plane to application plane in order to avoid SDN application layer attacks	IN/EX ST UST	AC/ PS	AP/CT NB
Mattos *et al*. [170]	AuthFlow is based on a RADIUS authentication system where an authenticator handles authentication requests between hosts and the RADIUS server and then provides a response to the OpenFlow controller that allows or denies traffic	Strengthening the network security on the basis of authentication and access control	IN/EX ST UST	AC/ PS	CT/DT SB

(*Continues*)

DOI: 10.1002/sec

Table XV. (Continued)

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Dangovas *et al.* [160]	AAA-SDN is an access control system to enforce SDN security. The access control system uses the controller to authenticate switches/hosts and to manage flows and user/host mobility	Preventing security issues of unauthorized access of hosts to SDN	IN/EX ST UST	AC/ PS	CT/DT SB
Zhu *et al.* [161]	SFA is a stateful forwarding abstraction in the SDN data plane that uses a forwarding processor to perform packet inspection	Enabling packet inspection that may require upper layers (L4–L7) information	IN/EX ST UST	AC/ PS	AP/ CTNB/ SB DT
Stoenescu *et al.* [162]	Sym-Net is a tool used to model basic stateful middleboxes that perform stateful checking in order to help in making contextual firewall decisions that depend on L2 to L7 information	Gathering all network layer information to have a global view to perform stateful inspections	IN/EX ST UST	AC/ PS	AP/ CTNB/ SB DT
Fayaz and Sekar [163]	FlowTest is a tool that allows testing firewall policies in SDN in order to make accurate decision regarding flow states as some SDN policies have no knowledge about a traffic state in the L4 and thus making an inappropriate decision by relying only on L2/L3 information	Testing stateful behaviors in the data plane to support the process of stateful inspection in L4 for a valid decision because states are indicated per TCP connections	IN/EX ST UST	AC/ PS	CT/ NBSB/ DT
Skowyra *et al.* [164]	Learning intrusion detection system is an OpenFlow-based NIDS. Learning intrusion detection system takes advantage of the OpenFlow SDN architecture, and anomalies are defined based on several characteristics: packets sent, position, time passed, size, etc.	Anomaly detection processes are built into the OpenFlow network to identify intrusions in embedded mobile devices	IN/EX ST UST	AC/ PS	CT/OF SB/DT AP

Table XVI. DoS attacks.

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Shin *et al.* [171]	Avant-Guard consists of limiting the flow requests sent to the control plane using a connection migration tool that removes failed TCP sessions at the data plane	Handling and mitigating DoS attacks as well as eliminating their impact on the network	IN/EX ST	AC	CT/ DTSB
Fonseca *et al.* [113]	CPRecovery provides network resilience through a transition from the failed primary controller to a backup controller	Quick and seamless backup to secondary controller in case of DoS attack on the primary controller	EX/ INST UST	AC	CT/DT/ SB
Naous *et al.* [173]	Ident++ protocol queries end hosts/ network on the path of a flow for additional information in order to make forwarding decisions. In this way end hosts participate in security enforcement	Dynamic flow table management and distributed control help in reducing the impact of DoS attack	EX/ INST UST	AC	CT/DT/ SB/ APNB/ OF

(*Continues*)

Table XVI. (Continued)

Countermeasure	Method/description	Purpose/challenge	Type of threat	Type of attack	SDN layer/ API
Braga *et al.* [174]	The system monitors OpenFlow switches registered to an NOX controller at regular intervals to retrieve traffic data for analysis	Achieving DDoS attack detection based on statistical information with self-organizing maps to classify traffic as normal or malicious	EX/ INST UST	AC/ PS	CT/SB/ AP/OF
Suh *et al.* [175]	CONA is based on an OpenFlow agent to detect DDoS as soon as a rate of requests from a given host to a content server exceeds a predefined value	OpenFlow agent is used to intercept content request messages in order to analyzed them so as to mitigate against DDoS attacks	EX/ INST	AC/ PS	CT/ DTSB/ AP
YuHunag *et al.* [176]	DDoS defender uses locator/ID separation protocol to detect and drop DDoS traffic based on traffic volume. Locator is changing and ID is unchanged that help in identifying authorized/malicious hosts	DDoS attack is detected based on traffic analysis. For a given host, if the rate of traffic exceeds a threshold, then, the traffic is dropped by the controller	EX/ IN/ST	AC/ PS	CT/ DTSB/ AP
Lim *et al.* [177]	DDoS blocking application counters botnet-based attack by monitoring flow metrics in the SDN controller to detect DDoS attacks and block botnets	Helping in detecting and blocking DDoS in case of botnet-based attacks	EX/ INST	AC/ PS	CT/ DTSB/ AP
Zaalouk *et al.* [178]	OrchSec utilizes network monitoring capabilities and SDN control to develop security applications	This architecture helps in detecting and overcoming DoS, worms, and DNS amplification attacks	EX/ INST UST	AC/ PS	CT/ DTSB/ AP
Schehlmann *et al.* [179]	COFFEE utilizes SDN capabilities to parse all network traffic in order to make more accurate detections and reduce the rate of false detections	Detect and mitigate botnets based on an OpenFlow detection approach. Botnets are used to perform DDoS attacks	EX/ INST	AC/ PS	CT/ DTSB/ AP
Namal *et al.* [143]	Switch mobility and secure change of IP addresses	Resilience against known TCP attacks to securely change IP address during OF switch mobility	EX/ INST	AC/ PS	SB
Liyanage *et al.* [166]	Secure SDMN uses a security layer to coordinate the communication between the controller and the OpenFlow switches	Extended version of OFHIP to tackle the issues in the software-defined mobile network control channel	EX/ INST	AC/ PS	SB
Benton *et al.* [120]	Assessment of OpenFlow vulnerabilities for DoS attacks	They showed that OpenFlow protocol and its related communication mechanisms between controller and switches should be thoroughly investigated	EX/ INST UST	AC/ PS	OF/SB

SDN, software-defined networking; SDMN, Software defined mobile networks; DOS, denial of service; DDoS, distributed denial of service.

data/control plane, API, and in the OpenFlow protocol. In [127–129], the authors discussed application failures and their risk in SDN implementations and respectively proposed FortNOX, ROSEMARY, and LegoSDN to deal with application issues. In terms of securely implementing the SDN, a number of solutions have been presented such as FRESCO [165], OFHIP [143], Secure SDMN [166], and Debugger for SDN [167]. For unauthorized access issues in SDN, several work have been presented like Securing Distributed Control [130], Byzantine-Resilient SDN [131], Authentication for Resilience [132], PermOF [137], Operation Checkpoint [114], SE-Floodlight [168], Flowtags [169], and AuthFlow [170]. In [113,143,166,171–179], analysis, enhancements, and solutions to counter DoS attacks have been presented. Other works, as shown in the following tables, deal with IP/ARP spoofing attacks (Table XII), SDN intrusions (Table XV), tampering (Table XIII), and anomalies/scanning attacks (Table XI). For clear visibility, we have classified the existing works depending on the security threats they can fully or partially address.

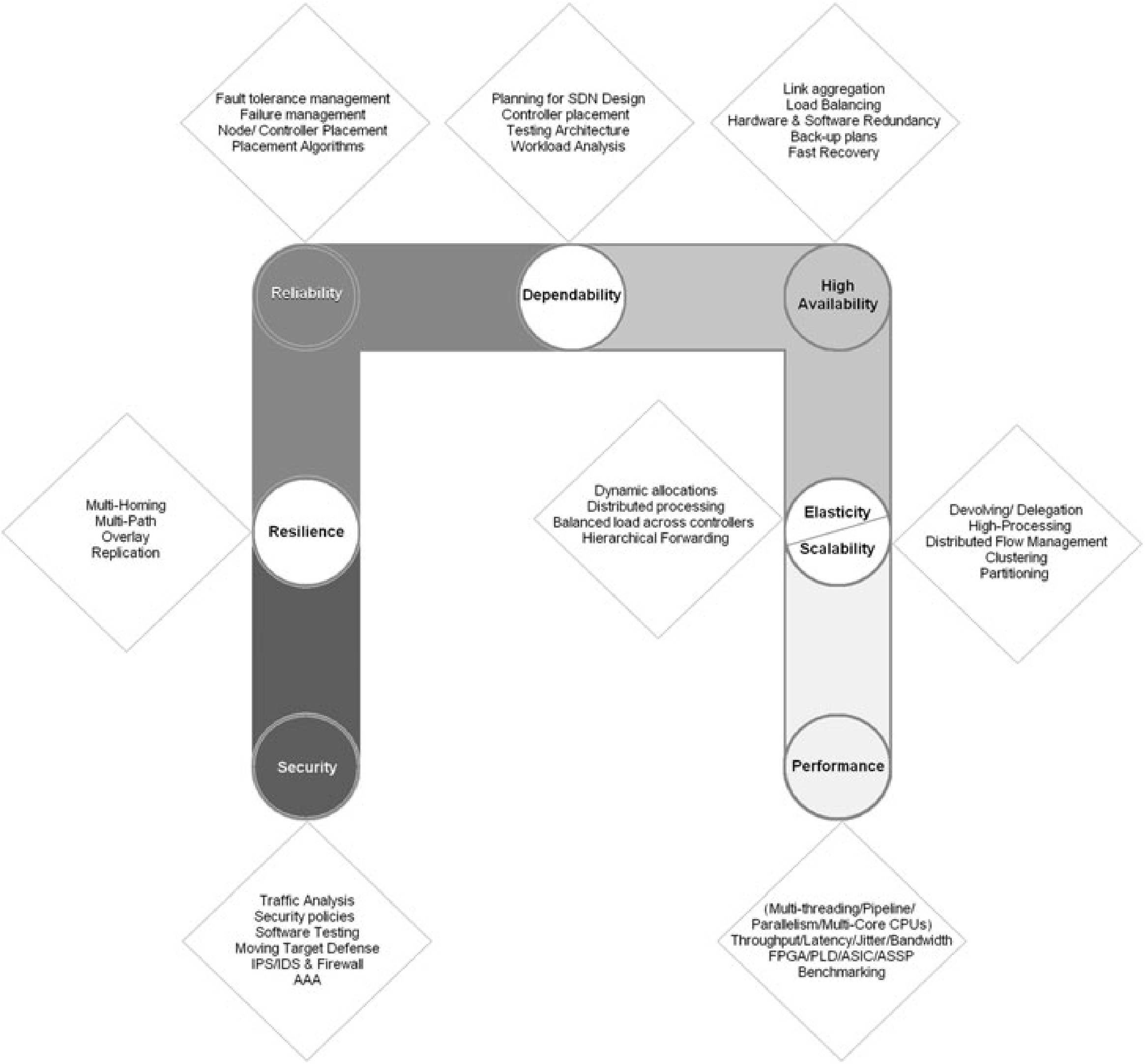

Figure 10. Major challenges and objectives for a carrier-grade SDN.

8. SUMMARY AND CONCLUSION

We have investigated the issues/challenges as well as the solutions for SDN in terms of scalability, elasticity, dependability, reliability, high availability, resiliency, security, and performance. As SDN is a new networking approach, several solutions to classical network problems have been revisited using this architecture, and many problems continue to be challenging. In this paper, we tried to simplify and explain each SDN issue and provided an overall view. It is important to note that the landscape of SDN-related issues changes according to the advances in SDN development. For instance, a new design modification or protocol introduced to SDN API might bring a new solution and at the same time incur new challenges or issues. Furthermore, many challenges in SDN still need further research attention such as the standardization of SDN components and adoption of new specific protocols designed to SDN in order to avoid inherited problems from the legacy networks. Research must focus more on the control plane to come up with novel solutions for controllers, which are the brains of the SDN architecture. The control plane is a point of failure of the whole network, and many security measures should be considered. Also, new HA and performance mechanisms should be implemented to abide by the SLA and deliver services accordingly. Service providers are willing to meet carrier-grade SDN requirements by ensuring scalability, reliability, and resilience of their infrastructure. However, by adopting an extensive range of sophisticated features into their networks, reliability of SDN becomes a challenging goal. In fact, this is challenging when it comes to sticking with service requirements and respect today's telecommunication and network commitments in terms of fast fault management, detection and recovery, prevention from unauthorized access and Dos attacks, high throughput, very low latency, and software upgrades and patches.

Figure 10 summarizes all SDN challenges to be considered when designing a resilient SDN network, which is still an open research area. To achieve resilience, the service providers must deploy secure components/policies and reliable mechanisms. As for dependability, both highly available and reliable infrastructures are required to ensure a continuous and reliable service delivery to end costumers. Elasticity and scalability are also two important goals in designing carrier-grade networks, but those can be only established in a highly available network with low latencies and high performance.

REFERENCES

1. Naudts B, Kind M, Westphal FJ, Verbrugge S, Colle D, Pickavet M. Techno-economic analysis of software defined networking as architecture for the virtualization of a mobile network. *Software Defined Networking* (*EWSDN*), *European Workshop on*. IEEE, 2012, pp. 67–72.
2. Benson T, Akella A, Maltz D. Unraveling the complexity of network management. *Proc*. *6th USENIX Symp*. *Networked Syst*. *Design Implement*. USENIX Association, 2009, pp. 335–348.
3. Floodlight controller, Floodlight documentation, for developers, architecture. [Online]. Retrieved from: http://www.projectfloodlight.org/floodlight/.
4. OpenDaylight: a Linux Foundation Collaborative Project, 2014. [Online]. Retrieved from: http://www.opendaylight.org.
5. Erickson D. The Beacon OpenFlow controller. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 13–18.
6. Casado M, Freedman M, Pettit J, Luo J, McKeown N, Shenker S. Ethane: Taking control of the enterprise. *Proceedings of the 2007 Conference on Applications*, *Technologies*, *Architectures*, *and Protocols for Computer Communications*. ACM, 2007, pp. 12.
7. McKeown N, Anderson T, Balakrishnan H, *et al.* OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 2008; **38**(2):69–74.
8. Open Networking Foundation Security Working Group. [Online]. Retrieved from: https://www.opennetworking.org/technical-communities/areas/services.
9. Doria A, Salim JH, Haas R, *et al*. Forwarding and Control Element Separation (ForCES) protocol specification, RFC 5810 (proposed standard), 2010. [online]. Available: https://datatracker.ietf.org/doc/rfc5810/.
10. Yang L, Dantu R, Anderson T, Gopal R. Forwarding and Control Element Separation (ForCES) framework, RFC 3746 (Informational), 2004. Available [online] : http://datatracker.ietf.org/doc/rfc3746/.
11. Farrel A, Vasseur JP, Ash J. A path computation element (PCE)-based architecture (No. RFC 4655). 2006.
12. Rodriguez-Natal A, Barkai S, Ermagan V, Lewis D, Maino F, Farinacci D. Software defined networking extensions for the locator/ID separation protocol, internet draft (experimental), 2014. Available [online]: http://wiki.tools.ietf.org/id/draft-rodrigueznatal-lisp-sdn-00.txt.
13. Lakshman TV, Nandagopal T, Ramjee R, Sabnani K, Woo T. The SoftRouter architecture. *Proceedings of the ACM Workshop on Hot Topics in Networks* (*HotNets*), San Diego, CA, USA, 2004.
14. Brocade Communications Systems, Network transformation with software-defined networking and Ethernet fabrics, California, USA. [online]. Available: http://www.brocade.com/downloads/documents/positioningpapers/network-transformation-sdn-wp.pdf, 2012.
15. Benzekki K, El Fergougui A, ElBelrhiti ElAlaoui A. A secure cloud computing architecture using homomorphic encryption. *International Journal of Advanced Computer Science & Applications* 2016; **1**(7):293–298.
16. Yu M, Rexford J, Freedman MJ, Wang J. Scalable flow-based networking with DIFANE. *SIGCOMM Computer Communication Review* 2010; **41**(4):351–362.
17. Tootoonchian A, Ganjali Y. HyperFlow: a distributed control plane for OpenFlow. *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*. USENIX Association, 2010, p. 3.
18. Cai Z. Maestro: achieving scalability and coordination in centralized network control plane, *Ph.D. dissertation*, Rice Univ., Houston, TX, USA, 2011.
19. Shalimov A, Zuikov D, Zimarina D, Pashkov V, Smeliansky R. Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central and Eastern European Software Engineering Conference in Russia*. ACM, 2013, p.1.
20. Rimal BP, Jukan A, Katsaros D, Goeleven Y. Architectural requirements for cloud computing systems: an enterprise cloud approach. *Journal of Grid Computing* 2011; **9**:3–26.
21. Armbrust M, Fox A, Griffith R, *et al.* A view of cloud computing. *Communication of the ACM, ACM* 2010; **53**(4):50–58.

DOI: 10.1002/sec

22. Dixit A, Hao F, Mukherjee S, Lakshman T, Kompella R. Towards an elastic distributed SDN controller. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN' 13*. ACM: New York, NY, USA, 2013; 7–12.
23. Dixit A, Hao F, Mukherjee S, Lakshman TV, Kompella R. ElastiCon: an elastic distributed SDN controller. *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2014, pp. 17–28.
24. Scott-Hayward S, O'Callaghan G,Sezer S. SDN security: a survey, IEEE *SDN for Future Networks and Services* (*SDN4FNS*), 2013, pp. 1–7.
25. Kloeti R. OpenFlow: a security analysis. April 2013. [Online]. Available: ftp://yosemite.ee.ethz.ch/pub/students/2012-HS/MA-2012-20signed.pdf.
26. Jarschel M, Oechsner S, Schlosser D, Pries R, Goll S, TranGia P. Modeling and performance evaluation of an OpenFlow architecture. Teletraffic Congress (ITC), 2011 23rd International, Sept 2011, pp. 1–7.
27. Cbench (Controller Benchmarker). [Online]. Available: http://www.openflow.org/wk/index.php/Oflops
28. Jarschel M, Lehrieder F, Magyari Z, Pries R. A flexible OpenFlow-controller benchmark. *Proc. EWSDN*, 2012, pp. 48–53
29. Tipper D. Resilient network design: challenges and future directions. *Telecommunication Systems* 2014; **56**(1):5–16.
30. Avižienis A, Laprie JC, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing, *Dependable and Secure Computing, IEEE Transactions on*. IEEE, 2004, pp. 11–33.
31. PatelP, Ranabahu AH, Sheth AP. Service level agreement in cloud computing, 2009.
32. Gude N, Koponen T, Pettit J, *et al.* NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review* 2008; **38**(3):105–110.
33. Maestro platform. [Online]. Available: http://code.google.com/p/maestro-platform/.
34. Trema, full-stack OpenFlow framework in Ruby and C. [Online]. Available: http://trema.github.com/trema/.
35. Manthena MPV, van Adrichem NL, van den Broek C, Kuipers F. An SDN-based architecture for network-as-a-service.
36. Pfaff B, Pettit J, Amidon K, Casado M, Koponen T, Shenker S. Extending networking into the virtualization layer. *Proc. HotNets*. 2009.
37. Wang A, Iyer M, Dutta R, Rouskas GN, Baldine I. Network virtualization: technologies, perspectives, and frontiers. *Journal of Lightwave Technology* 2013; **31**(4):523–537.
38. Manthena MPV. Network-as-a-service architecture with SDN and NFV: a proposed evolutionary approach for service provider networks, *Doctoral dissertation*, TU Delft, Delft University of Technology. 2015.
39. Curtis AR, Mogul JC, Tourrilhes J, Yalagandula P, Sharma P, Banerjee S. DevoFlow: scaling flow management for high-performance networks. *Comput. Commun. Rev.* 2011; **41**(4):254–265.
40. Mogul JC, Congdon P. Hey, you darned counters!: Get off my ASIC!. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, *ser. HotSDN '12*. New York, NY, USA. ACM, 2012, pp. 25–30.
41. Koponen T, Casado M, Gude N, *et al.* Onix: a distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, ser. OSDI'10*. USENIX Association: Berkeley, CA, USA, 2010; 1–6.
42. Hassas Yeganeh S, Ganjali Y. Kandoo: a framework for efficient and scalable offloading of control applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ser. HotSDN'12*. ACM: New York, NY, USA, 2012; 19–24.
43. Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R. On controller performance in software-defined networks. *Proc. 2nd USENIX Conf. Hot-ICE Netw. Serv.*USENIX Association, 2012, p. 10.
44. Sherwood R, Gibb G, Yap K-K, *et al.* FlowVisor: a network virtualization layer, Deutsche Telekom Inc. R&D Lab, Stanford, Nicira Networks, Tech. Rep., 2009.
45. Luo T, Tan H-P, Quan P, Law YW, Jin J. Enhancing responsiveness and scalability for OpenFlow networks via control-message quenching. *Proceedings of International Conference on ICT Convergence* (*ICTC*).IEEE, 2012, pp. 348–353.
46. Kempf J, Bellagamba E, Kern A, Jocha D, Takacs A, Skoldstrom P. Scalable fault management for OpenFlow. *Proceedings of IEEE International Conference on Communications* (*ICC*). IEEE, 2012, pp. 6606–6610.
47. Veisllari R, Stol N, Bjornstad S, Raffaelli C. Scalability analysis of SDN-controlled optical ring MAN with hybrid traffic. *Communications* (*ICC*), *IEEE International Conference on*. IEEE, 2014, pp. 3283–3288.
48. Park SH, Lee B, You J, Shin J, Kim T, Yang S. RAON: recursive abstraction of OpenFlow networks. *Proceedings of the Third European Workshop on Software Defined Networks* (*EWSDN*). IEEE, 2014, pp. 115–116.

49. Benzekki K, El Fergougui A, ElBelrhiti ElAlaoui A. Devolving IEEE 802.1X authentication capability to data plane in software defined networking (SDN) architecture. *Security and Communication Networks*, **9**(17), 4369–4377.
50. Voellmy A, Wang J. Scalable software defined network controllers. *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun*. ACM, 2012, pp. 289–290.
51. Krishnamurthy A, Chandrabose SP, Gember-Jacobson A. Pratyaastha: An efficient elastic distributed SDN control plane. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ser. HotSDN'14*. New York, NY, USA. ACM, 2014, pp. 133–138.
52. Bari MF, Roy AR, Chowdhury SR, *et al*. Dynamic controller provisioning in software defined networks. 9th International Conference on Network and Service Management, ser. CNSM'13, 2013.
53. Rajagopalan S,Williams D, Jamjoom H, Warfield A. Split/merge: system support for elastic execution in virtual middleboxes. *NSDI*. USENIX Association, 2013, pp. 227–240.
54. Fang L, Chiussi F, Bansal D, *et al*. Hierarchical SDN for the hyper-scale, hyper-elastic data center and cloud. *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, p. 7.
55. Aissioui A, Ksentini A, Gueroui A, Taleb T. Towards elastic distributed SDN/NFV controller for 5G mobile cloud management systems, IEEE. 2015.
56. Mueller J, Wierz A, Vingarzan D, Magedanz T. Elastic network design and adaptive flow placement in software defined networks. *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*. IEEE, 2013, pp. 1–6.
57. Vegad M. Elasticity in virtual middleboxes using NFV/SDN, *Doctoral dissertation*, Indian Institute of Technology, Bombay. 2015.
58. Longo F, Distefano S, Bruneo D, Scarpa M. Dependability modeling of software defined networking. *Computer Networks* 2015; **83**.
59. Wu J, Huang Y, Kong J, Tang Q, Huang X. A study on the dependability of software defined networks. *International Conference on Materials Engineering and Information Technology Applications (MEITA 2015)*. Atlantis Press. 2015.
60. Kreutz D, Ramos F, Verissimo P. Towards secure and dependable software-defined networks. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 55–60.
61. Jansen W. Cloud hooks: security and privacy issues in cloud computing. *System Sciences (HICSS), 44 the Hawaii International Conference on*. IEEE, 2011, 1–10.
62. Ahuja SP, Komathukattil D. A survey of the state of cloud security. *Network and Communication Technologies* 2012; **1**(2):66.
63. Link aggregation control protocol (LACP), http://www.cisco.com/c/en/us/td/docs/ios/122sb/feature/guide/gigeth.html, Mar. 2007.
64. EtherChannels, http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3550/software/release/12-113ea1/configuration/guide/3550scg /?swethchl.html.
65. IP multicast load splitting–equal cost multipath (ECMP), http://www.cisco.com/c/en/us/td/docs/ios/122sr/122srb/feature/guide/srbmpath.html.
66. Virtual router redundancy protocol (VRRP) version 3 for IPv4 and IPv6, http://tools.ietf.org/html/rfc5798, Mar. 2010.
67. Cisco hot standby router protocol (HSRP), https://www.ietf.org/rfc/rfc2281.txt, Mar. 1998.
68. Resilient packet ring (RPR), http://www.ieee802.org/17/docu-ments.htm.
69. Non-stop routing (NSR), http://www.cisco.com/c/en/us/td/docs/iosxml/ios/iprouteospf/configuration/15-e/iro-15-e-book/iro-nsr-ospf.html.
70. Graceful OSPF restart: non-stop forwarding (NSF), http://tools.ietf.org/html/rfc3623, Nov. 2003.
71. Stateful switch-over (SSO), http://www.cisco.com/c/en/us/td/docs/ios/12 0s/feature/guide/sso120s.html.
72. Ethernet automatic protection switching (EAPS), https://tools.ietf.org/html/rfc3619, Oct. 2003.
73. Ethernet ring protection switching (ERPS), http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cether/configuration/xe-3s/ce-xe-3s-book/ceg8032-ering-pro.html.
74. Fast re-routing (FRR), http://tools.ietf.org/html/rfc4090, May 2005.
75. Kim H, Santos JR, Turner Y, Schlansker M, Tourrilhes J, Feamster N. CORONET: fault tolerance for software defined networks. *Network Protocols (ICNP), 20th IEEE International Conference on*. IEEE, 2012, pp. 1–2.
76. Borokhovich M, Schiff L, Schmid S. Provable data plane connectivity with local fast failover: introducing OpenFlow graph algorithms. *Proc. 3rd Workshop Hot Topics Softw. Defined Netw*. ACM, 2014, pp. 121–126.
77. Heller B, Sherwood R, McKeown N. The controller placement problem. *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Software*

Defined Networking (*HotSDN*). ACM, 2012, pp. 7–12.

78. Sharma S, Staessens D, Colle D, Pickavet M, Demeester P. Enabling fast failure recovery in OpenFlow networks. *Design of Reliable Communication Networks* (*DRCN*), 8th International Workshop on the. IEEE, 2011, pp. 164–171.
79. Park H, Song S, Choi BY, Choi T. Toward control path high availability for software-defined networks. *Design of Reliable Communication Networks* (*DRCN*), *11th International Conference on*. IEEE, 2015, pp. 165–172.
80. Kuroki K, Fukushima M, Hayashi M, Matsumoto N. Redundancy method for highly available OpenFlow controller. *International Journal on Advances in Internet Technology* 2014; **7**(1 and 2).
81. Su Z, Wang T, Xia Y, Hamdi M. CheetahFlow: towards low latency software-defined network. Communications (ICC), *2014 IEEE International Conference on*. IEEE, 2014, pp. 3076–3081.
82. He K, Khalid J, Gember-Jacobson A, *et al*. Measuring control plane latency in SDN-enabled switches. *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015, Article No 25.
83. Williams D, Jamjoom H. Cementing high availability in OpenFlow with RuleBricks. *Proc. of ACM SIGCOMM Workshop on HotSDN*. ACM, 2013, pp. 139–144.
84. Desai M, Nandagopal T. Coping with link failures in centralized control plane architecture. *Proceedings of IEEE COMmunication Systems and NETworks* (*COMSNET*). IEEE, 2010, pp. 79–88.
85. Berde P, Gerola M, Hart J, *et al*. ONOS: towards an open, distributed SDN OS. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. ACM, 2014, pp. 1–6.
86. Hu Y, Wang W, Gong X, Que X, Cheng S. On reliability-optimized controller placement for software-defined networks. *China Communications* 2014; **11**:38–54.
87. Yao G, Bi J, Guo L. On the cascading failures of multi-controllers in software defined networks. *Network Protocols* (*ICNP*), *2013 21st IEEE International Conference on*. IEEE, 2013, pp. 1–2.
88. Ros FJ, Ruiz PM. Five nines of southbound reliability in software-defined networks. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. ACM, 2014, pp. 31–36.
89. Xiao P, Qu W, Qi H, Li Z, Xu Y. The SDN controller placement problem for WAN. *Communications in China* (*ICCC*), *IEEE/CIC International Conference on*. IEEE, 2014, pp. 220–224.
90. Capone A, Cascone C, Nguyen AQ, Sansò B. Detour planning for fast and reliable failure recovery in SDN with OpenState, *arXiv preprint arXiv*:*1411.7711*. 2014.
91. Pfeiffenberger T, Du JL, Arruda PB, Anzaloni A. Reliable and flexible communications for power systems: fault-tolerant multicast with SDN/OpenFlow. *New Technologies*, *Mobility and Security* (*NTMS*), *7th International Conference on*. IEEE, 2015, pp. 1–6.
92. Guan X, Choi BY, Song S. Reliability and scalability issues in software defined network frameworks. *Research and Educational Experiment Workshop* (*GREE*), *2013 Second GENI*. IEEE. 2013, pp. 102–103.
93. Dong M, Kimata T, Sugiura K, Zettsu K. Quality-of-experience (QoE) in emerging mobile social networks. *IEICE Transactions on Information and Systems* 2010; **E97-D**:2606–2612.
94. Jeong K, Kim J, Kim Y. QoS-aware network operating system for software defined networking with generalized OpenFlows. *Proc*. *IEEE NOMS*. IEEE, 2012, pp. 1167–1174.
95. Handigol N, Seetharaman S, Flajslik M, Johari R, McKeown N. Asterix: load-balancing as a network primitive. *Proc*. *9th GENI Eng*. *Conf*. (*Plenary*), 2010, pp. 1–2.
96. M. Ghobadi, S. Yeganeh, and Y. Ganjali, Rethinking end-to-end congestion control in software-defined networks. *Proc*. *11th ACM Workshop Hot Topics Netw*. ACM, 2012, pp. 61–66.
97. Al-Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera: dynamic flow scheduling for data center networks. *Proc*. *7th USENIX Conf*. *NSDI*. USENIX Association, 2010, p. 19.
98. Marcial F. Evaluating OpenFlow controller paradigms. ICN 2013, *The Twelfth International Conference on Networks*, 2013, pp. 151–157.
99. Syed AS, Jannet F, Maham F, Aamir S, Syed MA. An architectural evaluation of SDN controllers. *Communications* (*ICC*), *2013 IEEE International Conference on*. IEEE, 2013. pp. 3504–3508.
100. Cai Z, Cox AL, Ng TE. Maestro: a system for scalable OpenFlow control, Rice Univ., Houston, TX, USA, Tech. Rep. TR10-08, Dec. 2010.
101. Khattak ZK, Awais M, Iqbal A. Performance evaluation of OpenDaylight SDN controller.
102. Bianco A, Birke R, Giraudo L, Palacin M. OpenFlow switching: data plane performance. *Proc*. *IEEE ICC*. IEEE, 2010, pp. 1–5.
103. Tanyingyong V, Hidell M, Sjodin P. Improving PC-based OpenFlow switching performance. *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, New York, NY, USA, 2010, p. 13:1.

104. Luo Y, Cascon P, Murray E, Ortega J. Accelerating OpenFlow switching with network processors. *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, New York, NY, USA, 2009, pp. 70–71.

105. Rotsos C, Sarrar N, Uhlig S, Sherwood R, Moore AW. OFLOPS: an open framework for OpenFlow switch evaluation. *Proc. 13th Int. Conf.* PAM, 2012, pp. 85–95.

106. Zhou W, Li L, Chou W. SDN northbound REST API with efficient caches. *IEEE International Conference on Web Services* (*ICWS*), 2014, pp. 257–264.

107. Egilmez H, Dane S, Bagci K, Tekalp A. OpenQoS: an OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. *Asia-Pacific Signal Information Processing Association Annual Summit and Conference* (*APSIPA ASC*), 2012, pp. 1–8.

108. Xiong P, Hacigumus H, Naughton JF. A software-defined networking based approach for performance management of analytical queries on distributed data stores. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, *SIGMOD '14*, ACM, New York, NY, USA, Snowbird, Utah, USA, 2014, pp. 955–966.

109. Wendong W, Qinglei Q, Xiangyang G, Yannan H, Xirong Q. Autonomic QoS management mechanism in software defined network. *China Communications* 2014; **11**:13–23.

110. Cleder Machado C, Zambenedetti Granville L, Schaeffer-Filho A, Araujo Wickboldt J. Towards SLA policy refinement for QoS management in software-defined networking, in *IEEE 28th International Conference on Advanced Information Networking and Applications* (*AINA*).IEEE, 2014, pp. 397–404.

111. Openflow controller performance comparison. [Online]. Available: http://www.openflow.org/wk/index.php/Controller_Performance_Comparisons.

112. Liu L, Tsuritani T, Morita I, Guo H, Wu J. Experimental validation and performance evaluation of OpenFlow-based wavelength path control in transparent optical networks. *Optics Express* 2011; **19**:26578–26593.

113. Fonseca P, Bennesby R, Mota E, Passito A. A replication component for resilient OpenFlow-based networking. *Proc. IEEE Network Operations and Management Symposium* (*NOMS 2012*). IEEE, 2012, pp. 933–939.

114. Zhang X, Phillips C. Network operator independent resilient overlay for mission critical applications (ROMCA). *Communications and Networking in China COM. Fourth International Conference on*. IEEE, 2009, pp. 1–5.

115. Han J, Watson D, Jahanian F. Enhancing end-to-end availability and performance via topology-aware overlay networks. *Computer Networks* 2008; **52**(16):3029–3046.

116. Akella A, Maggs B, Seshan S, Shaikh A, Sitaraman R. A measurement-based analysis of multihoming. Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ser. SIGCOMM '03. New York, NY, USA. ACM, 2003, pp. 353–364.

117. Rohrer JP, Jabbarand A, Sterbenz JP. Path diversification: a multipath resilience mechanism. *Design of Reliable Communication Networks* (*DRCN*). *7th International Workshop on*. IEEE. 2009, pp. 343–351.

118. Li Y, Zhang Y, Qiu L, Lam S. SmartTunnel: achieving reliability in the Internet. *INFOCOM 2007*. *26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 830–838.

119. Chiu AL, Choudhury G, Clapp G, Doverspike R, Feuer M, Gannett JW, Xu D. Architectures and protocols for capacity efficient, highly dynamic and highly resilient core networks. *Journal of Optical Communications and Networking, IEEE/OSA* 2012; **4**(1):1–14.

120. Benton K, Camp LJ, Small C. OpenFlow vulnerability assessment. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 151–152.

121. Kreutz D, Ramos F, Verissimo P, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey, *arXiv preprint arXiv:1406.0440*, 2014.

122. Kreutz D, Ramos F, Verissimo P. Towards secure and dependable software-defined networks. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 55–60.

123. Li D, Hong X, Bowman J. Evaluation of security vulnerabilities by using ProtoGENI as a launchpad. *Global Telecommunications Conference* (*GLOBECOM 2011*). IEEE, 2011, pp. 1–6.

124. Shin S, Gu G. Attacking software-defined networks: the first feasibility study. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 165–166.

125. Smeliansky R. SDN for network security. *Science and Technology Conference* (*Modern Networking Technologies*) (*MoNeTeC*), *First International*. IEEE, 2014, pp. 1–5.

126. Schehlmann L, Abt S, Baier H. Blessing or curse? Revisiting security aspects of software-defined networking. *Network and Service Management* (*CNSM*), *10th International Conference on*. IEEE, 2014, pp. 382–387.

127. Porras P, Shin S, Yegneswaran V, Fong M, Tyson M, Gu G. A security enforcement kernel for OpenFlow networks. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 121–126.

128. Shin S, Song Y, Lee T, *et al*. Rosemary: a robust, secure, and high-performance network operating system. *Proceedings of the* 2014 *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 78–89.

129. Chandrasekaran B, Benson T. Tolerating SDN application failures with LegoSDN. *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 22.

130. Othman MM, Okamura K. Securing distributed control of software defined networks. *International Journal of Computer Science and Network Security* 2013; **13**(9).

131. Li H, Li P, Guo S, Yu S. Byzantine-resilient secure software defined networks with multiple controllers. *Communications* (*ICC*), *2014 IEEE International Conference on*. IEEE, 2014, pp. 695–700.

132. Yu D, Moore AW, Hall C, Anderson R. Authentication for resilience: the case of SDN, *ser*. *Security Protocols XXI*. Springer, 2013, pp. 39–44.

133. Schlesinger C, Story A, Gutz S, Foster N, Walker D. Splendid isolation: language-based security for software-defined networks. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 79–84.

134. Skowyra RW, Lapets A, Bestavros A, Kfoury A. Verifiably safe software-defined networks for CPS. *Proceedings of the 2nd ACM International Conference on High Confidence Networked Systems*. ACM, 2013, pp. 101–110.

135. Guha A, Reitblatt M, Foster N. Machine-verified network controllers. *ACM SIGPLAN Notices* 2013; **48**:483–494.

136. Ball T, Bjrner N, Gember A, *et al*. VeriCon: towards verifying controller programs in software-defined networks. *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2014, p. 31.

137. Wen X, Chen Y, Hu C, Shi C, Wang Y. Towards a secure controller platform for OpenFlow applications. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 171–172.

138. OpenFlowSec.org. Security-enhanced Floodlight. [Online]. Available: www.openflowsec.org.

139. Porras P, Cheung S, Fong M, Skinner K, Yegneswaran V. Securing the software-defined network control layer. *Proceedings of the Network and Distributed System Security Symposium* (*NDSS*), San Diego, California. 2015.

140. Foster N, Harrison R, Freedman MJ, *et al.* Frenetic: a network programming language. *ACM SIGPLAN Notices* 2011; **46**(9):279–291.

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

142. Mai H, Khurshid A, Agarwal R, Caesar M, Godfrey P, King ST. Debugging the data plane with Anteater. *ACM SIGCOMM Computer Communication Review* 2011; **41**(4):290–301.

143. Namal S, Ahmad I, Gurtov A, Ylianttila M. Enabling secure mobility with OpenFlow. IEEE Software Defined Networks for Future Networks and Services. IEEE, 2013.

144. Scott-Hayward S, Kane C, Sezer S. Operation Checkpoint: SDN application control. *22nd IEEE International Conference on Network Protocols* (*ICNP*). IEEE, 2014, pp. 618–623

145. Khurshid A, Zhou W, Caesar M, Godfrey P. VeriFlow: verifying network-wide invariants in real time. *ACM SIGCOMM Computer Communication Review* 2012; **42**(4):467–472.

146. Mehdi SA, Khalid J, Khayam SA. Revisiting traffic anomaly detection using software defined networking. In *Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg, 2011; 161–180.

147. Hand R, Michael T, Eric K. Active security. In *Twelfth ACM Workshop on Hot Topics in Networks (HotNets-XII)*. ACM: College Park, MD, 2013; 79–108.

148. Jafarian JH, Al-Shaer E, Duan Q. OpenFlow random host mutation: transparent moving target defense using software defined networking. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 127–132.

149. Kampanakis P, Perros H, Beyene T. SDN-based solutions for moving target defense network protection. *A World of Wireless*, *Mobile and Multimedia Networks* (*WoWMoM*), *IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.

150. Giotis K, Argyropoulos C, Androulidakis G, Kalogeras D, Maglaris V. Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN

environments. *Journal of Computer Networks* 2014; **62**:122–136.

151. Matias J, Tornero B, Mendiola A, Jacob E, Toledo N. Implementing layer 2 network virtualization using OpenFlow: challenges and solutions. *Software Defined Networking* (*EWSDN*), *2012 European Workshop on*. IEEE, 2012, pp. 30–35.
152. Yao G, Bi, Xiao P. Source address validation solution with OpenFlow/NOX architecture. *Network Protocols* (*ICNP*), *2011 19th IEEE International Conference on*. IEEE, 2011, pp. 7–12.
153. Yao G, Bi J, Feng T, Xiao P, Zhou D. Performing software defined route-based IP spoofing filtering with SEFA. *Computer Communication and Networks* (*ICCCN*), *23rd International Conference on*. IEEE, 2014,pp. 1–8.
154. Feng T, Bi J, Hu H, Yao G, Xiao P. InSAVO: intra-AS IP source address validation solution with OpenRouter. Proceedings *of the IEEE International Conference on Computer Communications* (*INFOCOM*). Orlando Bi J, Liu B, Wu J, Shen Y USA. IEEE, 2012, pp. 33–34.
155. Meyer C, Schwenk J. Lessons learned from previous SSL/TLS Attacks—a brief chronology of attacks and weaknesses. IACR Cryptology ePrint Archive, 2013, p. 49.
156. Hernan S, Lambert S, Ostwald T, Shostack. Threat modeling uncover security design flaws using the stride approach. MSDN Magazine-Louisville, 2006, pp. 68–75.
157. Casado M, Garfinkel T, Akella A, *et al*. SANE: a protection architecture for enterprise networks. *Usenix Security Symposium*. *ser*. *USENIXSS '06*, Berkeley, CA, USA, vol. 15. 2006.
158. Jia X, Wang JK. Distributed firewall for P2P network in data center. *ICCE-China Workshop* (*ICCE-China*). IEEE, 2013, pp. 15–19.
159. Katta NP, Rexford J, Walker D. Logic programming for software-defined networks. *Workshop on Cross-Model Design and Validation* (*XLDI*). Vol. 412. ACM, 2012.
160. Dangovas V, Kuliesius F. SDN-driven authentication and access control system. *The International Conference on Digital Information*, *Networking*, *and Wireless Communications* (*DINWC2014*). The Society of Digital Information and Wireless Communication, 2014, pp. 20–23.
161. Zhu S, Bi J, Sun C. SFA: stateful forwarding abstraction in SDN data plane. USENIX/?Open Networking Summit Research Track (ONS14), Santa Clara, USA 2014.
162. Stoenescu R, Popovici M, Negreanu L, Raiciu C. Symnet: static checking for stateful networks. *Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, 2013, pp. 31–36
163. Fayaz SK, Sekar V. Testing stateful and dynamic data planes with FlowTest. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. ACM, 2014, pp. 79–84.
164. Skowyra R, Bahargam S, Bestavros A. Software-defined IDS for securing embedded mobile devices. *High Performance Extreme Computing Conference* (*HPEC*).IEEE, 2013, pp.1–7.
165. Shin S, Porras P, Yegneswaran V, Fong M, Gu G, Tyso M. FRESCO: modular composable security services for software-defined networks. *Proceedings of Network and Distributed Security Symposium*, 2013.
166. Liyanage M, Ylianttila M, Gurtov A. Securing the control channel of software-defined mobile networks. *World of Wireless*, *Mobile and Multimedia Networks* (*WoWMoM*), *IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.
167. Handigol N, Heller B, Jeyakumar V, Mazieres D, McKeown N. Where is the debugger for my software-defined network?. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 55–60.
168. OpenFlowSec.org. Security-enhanced FloodLight. [Online]. Available: www.openflowsec.org.
169. Fayazbakhsh SK, Sekar V, Yu M, Mogul JC. FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. ACM, 2013, pp. 19–24.
170. Mattos DMF, Ferraz LHG, Duarte OCMB. AuthFlow: authentication and access control mechanism for software defined networking.
171. Shin S, Yegneswaran V, Porras P, Gu G. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2013, pp. 413–424.
172. Yao G, Bi J, Xiao P. Source address validation solution with OpenFlow/NOX architecture. *19th IEEE International Conference on Network Protocols* (*ICNP*). IEEE, 2011, pp. 7–12.
173. Naous J, Stutsman R, Mazieres D, McKeown N, Zeldovich N. Delegating network security with more information. *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*. ACM, 2009, pp. 19–26.
174. Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/?OpenFlow.

IEEE 35th Conference on Local Computer Networks (*LCN*). IEEE, 2010, pp. 408–415.

175. Suh J, Choi H, Yoon W, You T, Kwon T, Choi Y. Implementation of content-oriented networking architecture (CONA): a focus on DDoS countermeasure. *European NetFPGA Developers Workshop*, 2010.
176. YuHunag C, MinChi T, YaoTing C, YuChieh C, YanRen C. A novel design for future on-demand service and security. *Communication Technology* (*ICCT*), *12th IEEE International Conference on*. IEEE, 2010, pp. 385–388.
177. Lim S, Ha J, Kim H, Kim Y, Yang S. A SDN-oriented DDoS blocking scheme for botnet-based attacks. *Ubiquitous and Future Networks* (*ICUFN*), *2014 Sixth International Conf on*. IEEE, 2014, pp. 63–68.
178. Zaalouk A, Khondoker R, Marx R, Bayarou K. OrchSec: an orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions. *Network Operations and Management Symposium* (*NOMS*). IEEE, 2014, pp. 1–9.
179. Schehlmann L, Baier H. COFFEE: a concept based on OpenFlow to filter and erase events of botnet activity at high-speed nodes. *GI-Jahrestagung*, 2013, pp. 2225–2239.