Computer Networks 92 (2015) 189–207

Contents lists available at ScienceDirect

Computer Networks

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

Resilience support in software-defined networking: A survey

Anderson Santos da Silva [a,*], Paul Smith [b], Andreas Mauthe [c], Alberto Schaeffer-Filho [a]

[a] Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
[b] Safety and Security Department, AIT Austrian Institute of Technology, Austria
[c] School of Computing and Communications, Lancaster University, United Kingdom

ARTICLE INFO

Article history:
Received 7 May 2015
Revised 30 July 2015
Accepted 14 September 2015
Available online 20 October 2015

Keywords:
Software-defined networking
Network resilience
OpenFlow
Network challenges

ABSTRACT

Software-defined networking (SDN) is an architecture for computer networking that provides a clear separation between network control functions and forwarding operations. The abstractions supported by this architecture are intended to simplify the implementation of several tasks that are critical to network operation, such as routing and network management. Computer networks have an increasingly important societal role, requiring them to be resilient to a range of challenges. Previously, research into network resilience has focused on the mitigation of several types of challenges, such as natural disasters and attacks. Capitalizing on its benefits, including increased programmability and a clearer separation of concerns, significant attention has recently focused on the development of resilience mechanisms that use software-defined networking approaches. In this article, we present a survey that provides a structured overview of the resilience support that currently exists in this important area. We categorize the most recent research on this topic with respect to a number of resilience disciplines. Additionally, we discuss the lessons learned from this investigation, highlight the main challenges faced by SDNs moving forward, and outline the research trends in terms of solutions to mitigate these challenges.

1. Introduction

Computer networks are important for businesses and to support the operation of societally critical infrastructures, such as future (smart) electrical grids and government services. The growth in number and variety of end-to-end services that networks must support has led to a great deal of heterogeneity in the way networks are implemented, resulting in (i) complex protocols to handle the communication between network devices [1], (ii) difficult deployment of network policies by network administrators [2] and (iii) limited routing scalability [3–5]. Additionally, challenges to normal network operation, such as malicious attacks and prohibitive communication delay, demonstrate that computer networks have long-standing resilience requirements [6].

Resilience is the ability of the network to maintain an acceptable level of service when confronted with operational challenges [7]. A challenge is an atypical event that hinders the expected normal network operation [6,8]. In order to deal with a wide range of challenges, network resilience encompasses six major disciplines: security, survivability (including fault tolerance), performability, traffic tolerance, disruption tolerance and dependability [7]. When a network challenge arises, mitigation mechanisms should be activated, ideally without human intervention, to rapidly protect a network and the services it supports. However, the broad range of potential challenges that could befall a network requires sophisticated network (resilience) management systems that can detect and mitigate their effects [8]. Existing

* Corresponding author. Tel.: +555180152104.
E-mail addresses: assilva@inf.ufrgs.br (A.S. da Silva), paul.smith@ait.ac.at (P. Smith), a.mauthe@lancaster.ac.uk (A. Mauthe), alberto@inf.ufrgs.br (A. Schaeffer-Filho).

http://dx.doi.org/10.1016/j.comnet.2015.09.012
1389-1286/

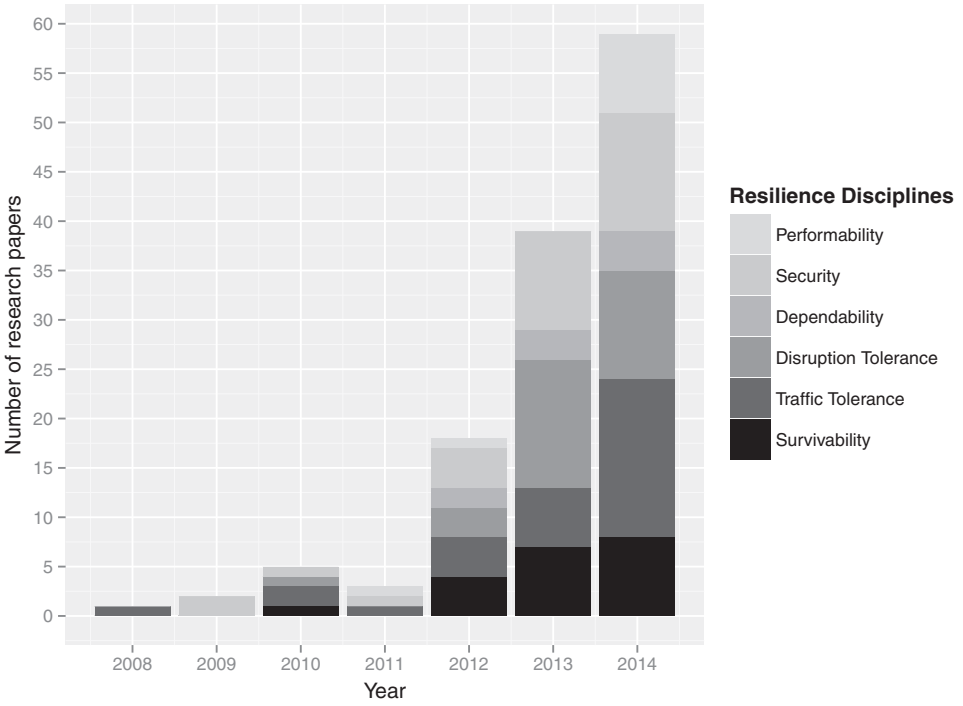

Fig. 1. Number of research papers included in this survey, according to their year of publication.

management systems have limitations, including a lack of flexibility with respect to challenge identification and mitigation, which has encouraged research that considers this problem in the context of new network architectures [9].

In both the research and industry communities, software-defined networking (SDN) [10] has recently gained significant attention. The main characteristic of the SDN architecture is that it decouples the implementation of network control logic from forwarding operations, thus enabling more flexible network control and management. In this context, a centralized *control plane* determines how forwarding devices, such as switches, will behave by configuring them using standardized protocols, such as OpenFlow [11]. The SDN architecture and the OpenFlow protocol, as its canonical implementation, offer (i) a comprehensive view of the network that is centralized in the *control plane*, (ii) high-levels of programmability of network applications, and (iii) fine-grained flow monitoring. These properties can be used to support the implementation of resilience mechanisms and help to minimize the complexity of managing them for network operators. Despite these benefits, new resilience challenges can arise because of the use of SDN, e.g., with respect to the fault tolerance of the control plane; research into addressing these issues is currently a major concern.

This paper presents a survey on the support for network resilience in software-defined networking. Research into this topic has recently intensified, as illustrated in Fig. 1, which summarizes the number of research papers addressing resilience aspects in SDN included in this survey, according to their year of publication. We organize the literature surveyed using the resilience taxonomy proposed by Sterbenz et al. [7], thus enabling a reliable categorization of the existing research efforts on SDN. Our survey discusses aspects such as existing solutions for resilience challenges, current open issues and research trends in this field. The aim of the survey is to present to the reader a comprehensive and structured view of network resilience in the SDN spectrum, and how resilience aspects are supported in these architectures.

We have observed that solutions related to fault management, infrastructure planning, routing and security applications, network measurement and anomaly detection are frequently used to address resilience challenges in the SDN context. However, we have identified several open issues in this research space, including the protection of the communication channel between network controller and forwarding devices; adequate support for sophisticated QoS solutions to enhance performability; and the need to detect novel malicious attacks targeting network devices.

The remainder of this paper is organized as follows. Section 2 presents necessary background material on SDN and network resilience. Section 3 discusses the proposed categorization of SDN efforts, with respect to different resilience disciplines. Section 4 shows a summary of the main research topics studied, topics already solved and others under investigation. Finally, Section 5 presents the concluding remarks.

2. Background

This section discusses the basic concepts and terminology used in this work. In particular, software-defined networking and network resilience are contextualized.

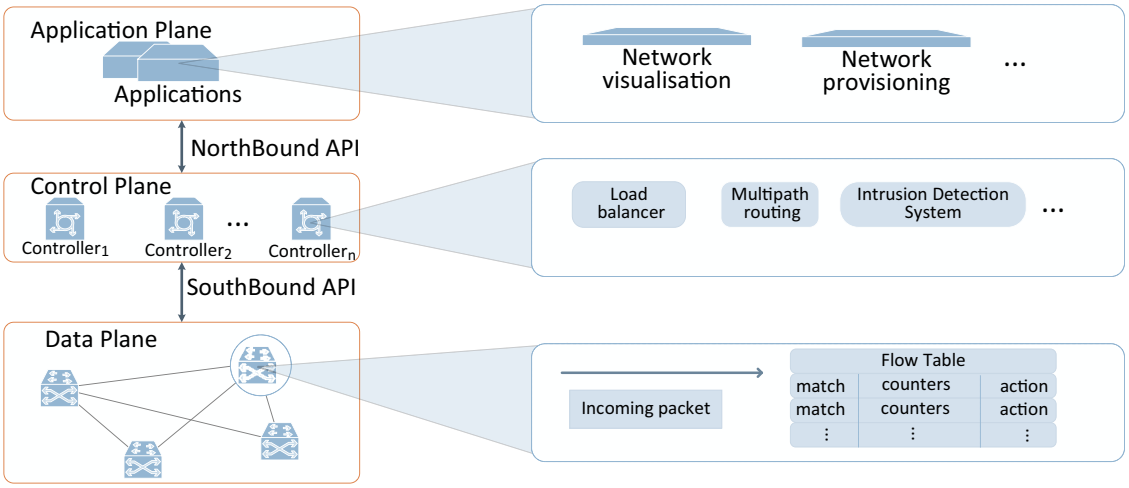

Fig. 2. SDN architecture: conceptual planes and communication interfaces.

2.1. Software-defined networking

Software-defined networking (SDN) is an architecture for computer networks aimed at decoupling the network control functions (*control plane*) from the forwarding devices (*data plane*) [10]. The *control plane* is responsible for determining the network control logic, such as implementing routing protocols. The aim of the SDN architecture is to simplify the deployment of new control plane functions, such as routing strategies, when compared to traditional networks [12,13], in which the control and data planes are more tightly coupled and typically operate in an entirely distributed fashion.

The SDN architecture defines three conceptual planes and communication interfaces as depicted in Fig. 2:

- The *application plane* is responsible for executing applications that run over the network infrastructure. Generally, these applications perform modifications regarding network aspects, such as network policies and routing behavior, with some degree of human intervention [12]. Examples of network applications deployed in this plane are network visualization, path reservation and network provisioning.
- The *control plane* defines control logic, such as routing schemes. Additionally, the control plane can manage the information collected by switches at the *data plane*, such as flow statistics, to orchestrate the traffic behavior. This plane has a global network view, being able to offer mechanisms for fault diagnosis, make decisions over current traffic distributions and enforce QoS policies. Usually, the *control plane* is physically distributed into *controller* devices, but logically centralized [14].
- The *data plane* includes the devices that are responsible for forwarding data, which are generally referred to as *switches*. An OpenFlow switch offers the notion of programmable flow tables, *i.e.*, tables that define an action for each packet associated with a specified flow. A flow table can be dynamically configured by the *control plane*. When a new packet arrives in a given switch it can be (i) dropped; (ii) flooded through all output ports; (iii) sent to a specific output port; or (iv) sent to the network controller [15]. For every flow the switches involved in this communication store statistical information that can be accessed by the *control plane*.

Furthermore, the communication between the different planes occurs through the following interfaces:

- *Northbound API*: Implements the communication interface between the *control plane* and the *application plane*. This API enables the programmability of the network controller by exposing network data abstractions to the *application plane*. Currently, the most used protocol for this communication is REST (REpresentational State Transfer).
- *Southbound API*: Implements the communication interface between the *control plane* and the *data plane*. Through this interface it is possible for the *control plane* to configure switches with forwarding actions according to received notifications of incoming packets from the *data plane* [16]. This is typically standardized and implemented by the OpenFlow protocol [11,17].

It can be seen that through these interfaces the SDN architecture introduces a great deal of flexibility in flow management, impacting directly in areas such as security, traffic management and performability [18,19]. Also, SDN has the potential to reduce the cost of network deployment, because simplified data plane switches are relatively inexpensive components, when compared to more complex routers [20]. Furthermore, OpenFlow has proven to be ideal for the development of prototype network applications [21]; based on this success, research in this field has increased [22]. These characteristics generated enthusiasm in both industry and academia. Many surveys covering historical aspects, architecture and challenges related to SDN have been published and further discussions can be found in [12,23–25].

2.2. Network resilience

Computer networks support many societally critical functions such as business transactions, military operations and electricity supply. However, a wide range of challenges

such as malicious attacks, operational overload and misconfigurations can occur, which could result in failures. Additionally, networked systems can include hardware and software faults that could similarly result in failures, if triggered. Approaches to network resilience aim to protect the network and overcome a degradation in the performance of services when confronted with challenges and faults.

2.2.1. General principles

There are a number of different ways in which a network can fail to provide a desired level of service, such as a given end-to-end delay or level of availability. Failures can be caused by so-called *challenges*, such as a malicious attack or natural disaster, or a fault. For example, critical failures resulting from natural disasters like Hurricane Katrina [26], in 2005, destroyed much of the network infrastructure on the east coast of the United States. The damage was such that communication links were interrupted and the electrical distribution system was compromised. Furthermore, our increasing dependence on network infrastructures has attracted the attention of cyber criminals. These individuals aim to disrupt the operation of large corporations or even nations through cyber-attacks that are targeted at the communication infrastructure [27]. In this case, a failure is the result of deliberate malicious activities that can compromise a target network service.

The variety of ways that networks can be challenged creates the need for a wide range of resilience mechanisms, which should be deployed across systems, network layers and infrastructures, as necessary. Unfortunately, an ideal resilience system that is capable of protecting the network from any challenge in any environment is difficult to achieve and expensive. A trade-off between the complexity of the mechanisms and their cost exists, and this restriction defines what can be deployed in practice [28]. Consequently, prohibitive costs can make ideal resilience solutions, such as fully fault tolerant systems, infeasible [20].

In general, a number of stages can be implemented to ensure the resilience of networks [7,29]. Initially a set of *defense* mechanisms should be deployed that address the known challenges a network may face; these might include firewalls or redundant network paths, for example. In some cases, these defense measures will fail, e.g., because new challenges emerge or they are insufficiently provisioned. Consequently, it is important to *detect* challenges and service degradation, and subsequently *diagnose* the root cause of a challenge. Using the outcomes from these stages, mechanisms to *remediate* the challenge can be used to adapt the system operation, in order to ensure continued or graceful degradation of the service. For example, suspicious network traffic can be subjected to deep-packet inspection (a form of detection and diagnosis), while malicious traffic can be blocked (a remedial action). Finally, when a challenge has abated, the network should *recover* to normal operation by disengaging remediation mechanisms, for example.

2.2.2. Resilience disciplines

In this section, we discuss the resilience disciplines that are related to networked systems. According to Sterbenz et al. [7], a number of existing disciplines address aspects of network resilience, which can be placed into two categories: *(i)* disciplines that provide mechanisms to address different classes of challenges and faults; and *(ii)* those specifying measurable properties that indicate the resilience of a network. We summarize these disciplines:

- *Survivability*: is a superset of *fault tolerance*, which addresses small numbers of random uncorrelated faults, by considering *numerous correlated failures* that could be caused by challenges such as malicious attacks and large-scale natural disasters. While *redundancy* is often considered sufficient for fault tolerance, ensuring the survivability of networks requires approaches to *diversity* to be implemented.
- *Traffic tolerance*: enables the network to tolerate unusual traffic load without interrupting its operation. It deals with legitimate traffic management, *e.g.*, *flash crowds*, largely via traffic engineering mechanisms. However, Distributed Denial of Service (DDoS) attacks can behave similarly to legitimate traffic, thus creating the need to identify and treat this type of traffic in a specific manner.
- *Disruption tolerance*: enables the network to tolerate weak and episodic connectivity that is typical of mobile and wireless networks. Approaches to disruption tolerance can include error correction schemes, multi-path routing and in extreme cases (e.g., for mobile ad hoc networks) store-carry-forward schemes. Often in this context, there are *energy* trade-offs that need to be addressed.
- *Dependability*: quantifies the reliance that can be placed on the service delivered by a system. Consequently, this definition encompasses concepts related to *availability* – an indicator of whether a service will be present when requested – and *reliability* – a measure of continued operation for a specified period of time. Also, dependability relates to measures of *safety*, *integrity* and *maintainability*.
- *Security*: is a property and set of measures that relate to unauthorized access to a networked system, and includes notions of self-protection. It includes concepts such as *confidentiality*, *nonrepudiality* and *AAA* (*auditability*, *authorizability*, *authenticity*). Additionally, security properties intersect with the dependability concepts of *availability* and *integrity*, with subtly different semantics—for security, these properties typically relate to *information assets*, rather than services.
- *Performability*: metrics describe the network's ability to deliver the performance required by its users; these requirements are normally expressed in quality of service (QoS) agreements. Typical performability metrics include delay, jitter, throughput and goodput, for example.

3. State-of-art in SDN resilience

This section discusses the main research efforts that have addressed resilience aspects in the context of SDN. The methodology employed to produce this survey is as follows. First, a total of 142 research papers and technical reports on SDN and resilience were gathered, based on criteria such as date and relevance. Second, the publications were categorized into the different resilience disciplines that are proposed by Sterbenz et al. [7] (see Section 2). Third, these works were arranged into one or more SDN planes (Fig. 2), in order to provide a high-level view of the intended resilience

Table 1
SDN research efforts on Fault Tolerance and Survivability.

SDN Planes	Fault Tolerance	Survivability
Application plane	Reitblatt et al. [30]	Chandrasekaran et al. [31]
	Heller et al. [32]	
	Canini et al. [33]	
	Scott et al. [34]	
Control plane	Jain et al. [35]	Williams et al. [36]
	Botelho et al. [37]	Liu et al. [38]
	Jian et al. [39]	Kempf et al. [40]
	Fonseca et al. [41]	Chandrasekaran et al. [31]
	Tootoonchina et al. [42]	Muller et al. [43]
	Zhang et al. [44]	
Data plane	Jain et al. [35]	Liu et al. [38]
	Botelho et al. [37]	Kempf et al. [40]

support over the different planes. In the following, for each resilience discipline, we discuss the main challenges and the solutions used to protect the network in SDN environments. Despite our efforts to accurately categorize the publications within the several resilience disciplines, we acknowledge that in some cases the same publication could be classified differently according to the assessment of the reader.

3.1. Fault tolerance and survivability

A natural concern about the SDN architecture is the survivability of the network controller, forwarding devices in the data plane and applications that monitor and change the network operation. To achieve the protection of these components, survivability encompasses the treatment of *fault tolerance* in the face of uncorrelated failures that are caused by faults, and *multiple correlated failures* that are caused by natural disasters and attacks, for example. Table 1 presents the major research efforts addressing survivability in SDN. In the following we discuss these works in detail.

Fault tolerance: is the ability of a system to provide continued operation or degrade gracefully in the presence of faults. The classical approach to fault tolerance is to introduce redundancy into the system, *e.g.*, through the use of replicas to protect critical components in the network [45]. In the SDN context, redundancy can be used to (i) protect the network controller from service failures, (ii) to protect the forwarding devices and communication links from link disruption, and (iii) to protect network applications from misconfiguration [44]. Despite the benefits of fault tolerance, a fully fault tolerant system brings complex issues that are related to the management of replicated components and equipment costs. SDN can help to address these issues because its architecture can accommodate the control of complex network functions. For example, the *control plane* can be used to orchestrate the behavior of active and redundant devices, eliminating the need for new devices to perform such tasks [41].

Jain et al. [35] relate experience with the use of fault tolerant approaches to address network outages and failures with B_4, a private WAN that connects the Google's data center. They used OpenFlow to manage individual *switches* that implement several fault tolerance tasks, such as multipath routing regarding flow priority and dynamic reallocation of bandwidth when link failures occur. The redundancy is achieved using software replicas to protect individual and control processes in the *control plane*. These replicas are placed on different physical servers and, in case of network faults, a consensus algorithm can be used to elect the replica that will assume the demand of the network. All the configurations and communications used to protect the network are assisted by an SDN-based architecture.

Botelho et al. [37] treat the consistency between a network controller and their redundant backups, focusing on performance aspects. The main conclusion of the authors is that strict consistency and fault tolerant systems can operate with acceptable performance. However, issues related to latency were shown to limit the responsiveness of the system. Jian et al. [39] confirm this observation and show that the performance of the network controller can be critical to link failure detection, even in fault tolerant systems.

We understand that a large number of applications that are related to fault tolerance running in the *control plane* can reduce its capacity to process network traffic. A solution to tackle this limitation is moving the majority of these applications to the *application plane*, which is more scalable and suitable for running these tasks mainly because this plane is designed exclusively to support the execution of general SDN applications. Using the programmability offered by the SDN architecture, Reitblatt et al. [30] propose FatTire, a language for writing fault-tolerant systems. The idea behind this approach is to offer an abstraction for network paths, such that the forwarding behavior of network packets can be orchestrated though regular expressions that are converted into primitive switch rules. This is an *application plane* solution, and can be updated according to the need of new resilience requirements. Furthermore, Fonseca et al. [41] investigate the redundancy of network controllers. Related to this is the work by Tootoonchian et al. [42], which deals with the distribution of controller state over the network, in order to offer a logically centralized network controller.

A recent concern involves the programmability offered by SDN and the challenges related to software debugging. In our understanding these issues also bear some relationship to Fault Tolerance. Heller et al. [32] discuss the overall problem space regarding troubleshooting in SDN but the authors do not propose any system or framework. Scott et al. [34] also deal with this aspect and in addition propose a troubleshooting system (called STS), which aims to alleviate the time-consuming nature of debugging by eliminating events that are not causally related to the source of a failure. Further, Canini et al. [33] are also concerned with troubleshooting

issues, and apply model-checking techniques to represent the state space of the network. This strategy is useful to detect design flaws, a frequent type of software bug.

Survivability: multiple, uncorrelated failures can be unpredictable and difficult to diagnose, and in this case redundancy may not be enough. A typical strategy to handle this kind of failures is *diversity*, *i.e.*, to use a set of distinct resilience schemes to determine and treat the source of failure. This increases the success of detection/mitigation schemes because a wide-range of network challenges can be addressed.

For example, *diversity* can be typically employed to withstand catastrophic faults, such as natural disasters. Muller et al. [43] highlight that even if the *control* and *data planes* are compromised, different placement strategies of the network controller, path diversity and distinct recovery mechanisms can be used to ensure that the network can still function. Note that similar techniques can be used not only to improve aspects related to survivability but also to disruption tolerance.

Chandrasekaran et al. [31] discuss how to handle challenges at the *application* and *control planes*. Their focus is to treat Byzantine failures, fail-stop crashes and other uncorrelated failures. They propose two abstractions for improving the network controller availability and diagnose network application faults: a module used for fault isolation and another to deal with network transactions. The joint operation of these components enables the orchestration of high-level applications, such as fault alerts and the specification of policies to enforce actions when failures occur. The proposed framework still enables the distribution of network events to different SDN applications, in order to tolerate multiple, uncorrelated failures.

SDN is a suitable environment to investigate *diversity* of automatic recovery mechanisms. For example, mechanisms used to plan for failure can be placed in distinct network controllers [36], and *control plane* applications can be used to ensure path reservation in the *data plane* [38]. Related to these ideas, Kempf et al. [40] highlight that fault management in SDN cannot be left to be fully implemented in the *control plane*, and instead delegates these tasks to the OpenFlow switches. They advocate that some control functions, such as connectivity monitoring, can be placed in the *data plane*. Consequently, *diversity* can be attained if different resilience mechanisms are implemented in these switches.

3.2. Dependability

Even if a system fails, it can be considered *dependable* if failures occur with an expected probability. Thus, dependability quantifies the reliance that can be placed on the service delivered by a system. In this sense, dependability encompasses concepts involving *reliability* and *maintainability*. It is also sometimes related to *safety*, *availability* and *integrity*.[1] Table 2 categorizes the major research efforts related to dependability in SDN. These are discussed in the following. Note that the research papers discussed in this section are all related to the *control plane*.

Table 2
SDN research efforts on dependability.

SDN planes	Reliability	Availability
Application plane	N/A	N/A
Control plane	Veisllari et al. [46] Deguo et al. [48] Ros et al. [50] Santos et al. [52] Dixit et al. [53]	Heller et al. [47] Hock et al. [49] Beheshti et al. [51]
Data plane	N/A	N/A

Reliability: Deguo et al. [48] state that the *data plane* cannot detect failures in the *control plane* and the number of control messages lost when a failure occurs can compromise the forwarding behavior of the network. A solution broadly accepted to deal with this challenge is to delegate some network control logic to the forwarding devices, such as rule cloning and multipath support [54]. A single point of failure is another reason that contributes to the decentralization of the *control plane* [52,53].

Ros et al. [50] investigate the controller placement problem with respect to network reliability. The authors propose a metric called *k-terminal-reliability*, which is the probability of having at least one operational path in the network. The authors can optimize the solution for reliability by formulating the controller placement problem as a graph optimization problem, and inserting the *k-terminal-reliability* as a restriction when searching for the optimal solution.

The reliability of the *data plane* can be related to switches and link state. Metrics such as communication delay, throughput and latency are frequently used as indicators of reliability, as these represent expected values that can or cannot be satisfied at any time.

Availability: this is the probability of a system to be in a correct state in a given instant. New metrics that are related to this issue are not currently the focus of intense study, however the controller placement problem is a research question that is related to availability. This problem investigates (i) how many controllers are needed to control a given network and (ii) what is the best place to position the controller regarding metrics of availability [47]. The most commonly used metrics measure the average-case latency, worst-case latency and the maximization of number of nodes with latency bound. Hock et al. [49] propose more elaborate metrics, using latency during controller failures, load imbalance and inter-controller latency. Beheshti et al. [51] propose metrics such as the number of protected switches (switches that can use backup links for the control traffic) and the number of unprotected switches.

3.3. Security

Historically the deployment of complex security functions (*e.g.*, intrusion detection systems and firewalls) has required the installation of dedicated security appliances. For some organizations the costs and management issues related to these deployments can be prohibitive. Additionally, in

[1] Safety and maintainability are rather general properties and to the best of our knowledge there are no research efforts in SDN solely concentrated on these fields. Integrity is related to security and will be discussed in the next section.

Table 3
SDN research efforts on security.

SDN planes	Confidentiality	Availability	Integrity	Nonrepudiality
Application plane	N/A	Chen et al. [55] Wang et al. [56] Seeber et al. [57] Tasch et al. [58]	N/A	N/A
Control plane	Benton et al. [59] Schehlmann et al. [63] Kreutz et al. [64] Porras et al. [66] Anwer et al. [69] Fayazbakhsh et al. [71] Naous et al. [73] Silva et al. [75] Ballard et al. [77] Schlesinger et al. [79]	Li et al. [60] Zaalouk et al. [61] Schehlmann et al. [63] Mazieres et al. [67] Chen et al. [55] Wang et al. [56] Seeber et al. [57]	Zaalouk et al. [61] Schehlmann et al. [63] Ye et al. [65] Collings et al. [68] Hu et al. [70] Xing et al. [72] Kampanakis et al. [74] Jafarian et al. [76] Smeliansky et al. [78] Abaid et al. [80] Benton et al. [59] Shin et al. [81] Kumar et al. [82] Li et al. [83] Qazi et al. [84] Son et al. [85]	Nayak et al. [62]
Data plane	Kreutz et al. [64] Shin et al. [81] Naous et al. [73] Qazi et al. [84]	Chen et al. [55] Wang et al. [56] Seeber et al. [57]	Hu et al. [70] Xing et al. [72] Smeliansky et al. [78]	N/A

traditional networks the lack of a centralized control of these security functions can further complicate their deployment [61]. In contrast, SDN enables the implementation of applications that have the ability to support similar security functions in a much more flexible manner, and it offers a suitable place for the implementation of more accurate, reliable and efficient security solutions. Table 3 presents the major research efforts addressing security in SDN. In the following we discuss these in detail.

Availability: Schehlmann et al. [63] state that the availability of the network controller can compromise the correct operation of network functions. To address the availability aspects of the network controller with respect to security, Li et al. [60] propose a novel SDN architecture based on BFT (Byzantine Fault Tolerant) [67] mechanisms to withstand malicious attacks on the *control plane*. The authors state that a distributed control plane can assist in protecting the network mainly because a single point of attack is avoided. Also, the authors highlight that the additional protection strategies, such as the use of BFT, can ensure the correct operation of critical network functions (*e.g.*, flow tables updates).

With a distributed *control plane*, it is natural to consider the distributed placement of security applications. However, this can increase the communication delay in security traversal routing, *i.e.*, the traversal of a given flow through secure devices to enforce security inspection. Chen et al. [55] address the security traversal problem with shortest path solutions, including the ability to dynamically select the optimal security traversal path. Cloud environments, for example, can benefit from solutions of this type, since security issues related to communication are critical [56,57].

It is possible that not only the network controller but also the network applications will be target of attacks. In line with this, the availability of security applications is addressed by Tasch et al. [58]. The authors state that even consolidated applications such as RESONANCE [62] can have security problems, such as identity spoofing and repudiation when TLS is not available to protect the control communication.

Integrity: several works suggest that firewalls are more concerned with (the integrity aspects of) security. Despite all functions that these systems can perform, their main goal is to maintain the integrity of the communication link and network devices, *i.e.*, protect the network from illegitimate attempts to gain access to its services [68]. Firewalls are an example of network application that can be fully assisted by SDN because: (i) the need of additional middleboxes to enforce policies in the network is reduced because this functionality can be placed in the SDN *control plane*; (ii) the *control plane* has a comprehensive view of the network; and (iii) the management of heterogeneous devices is abstracted by the *control plane*. Also, Qazi et al. [84] present a policy security monitoring layer for efficient middlebox-specific *traffic steering*. Another work that goes in this direction is proposed by Son et al. [85].

Resolution of firewall policy violations and conflicts are addressed by Hu et al. [70]. The authors propose the *FlowGuard* framework to monitor and check network flows in order to detect firewall policy changes when the network state is updated. However, they identified the following challenges with respect to the implementation of firewalls in SDN: (i) as the network state is dynamically changed, new configurations are frequent and simple packet-in monitoring will not be effective for detecting flow policy violations; (ii) the *set-field* actions of the OpenFlow protocol enable the dynamic change of packet headers, creating an opportunity for malicious users to attack the network; (iii) as OpenFlow enables the use of *wildcards* to match only partial header fields of packets in these security policies, the elimination of a flow policy can affect benign traffic; (iv) the *data plane* is unable to

monitor flow status, depending heavily on the *control plane*, thus it is challenging to perform stateful packet inspection.

Intrusion Detection Systems, such as Snort,[2] have been used to protect the integrity of the network in traditional environments. SnortFlow [72] is an extension to Snort with SDN capabilities that enables the detection of intrusions and malicious activities in cloud environments. Detection mechanisms are traditionally based on machine learning [86], signatures [87] and entropy [88]. Shin et al. [81] group several security needs and deliver a complete framework for security implementation, sharing and composition of detection modules and mitigation in an SDN. This effort enables the evaluation of optimal mechanisms to protect the network, being able to detect and manage malicious activities. Another example of IDS with these responsibilities is presented by Kumar et al. [82]. Solutions based on packet classification using SDN are presented by Smeliansky et al. [78] and the detection of malware is discussed in the work of Abaid et al. [80]. Further, the PROTOGENI framework is presented by Li et al. [83] and serves as a tool for creating and evaluating different types of network attacks.

However, other solutions to protect the network from intrusions and malicious attacks are possible. Kampanakis et al. [74] advocate the use of *moving target defense* (MTD) in order to protect network services when malicious traffic tries to compromise their integrity. The authors conclude that SDN makes the implementation of MTD techniques more practical, customizable and easier to deploy. Jafarian et al. [76] present a study related to this problem and propose a technique named *random host mutation*, *i.e.*, the path between source and target is randomized to avoid the effects of malicious traffic.

Security threats can also compromise the integrity of configuration messages sent by the *control plane* to the *data plane* by modifying or introducing errors in their content [65]. A simple solution for this challenge is to use encryption protocols, such as TLS. Benton et al. [59] state that the biggest concern related to security in SDN is the protection of the communication between the *control plane* and the *data plane*. The authors point out that the TLS protocol in the OpenFlow specification may sometimes be used incorrectly, because in order to put TLS in practice, the network operator must achieve security certificates for each of the devices involved in the communication and manually configure each of them. In contrast, to use plain text communication without any encryption, the network operator only needs to configure the network controller address in the *data plane*. Thus, the difficulty in deploying TLS can discourage its use.

Confidentiality: related to the issues discussed above, another challenge pointed out in the work of Benton et al. [59], and uniquely related to SDN, is called the *listener mode*. This existing functionality in many *switches* allows the establishment of a data connection in a pre-configured port without authentication. Although it is used primarily for debugging reasons, this connection can be used to modify rules in switches and discover information about the network. If TLS is not used correctly, an attacker can intercept packets and perform network discovery based on communications observed between the *control plane* and the *data plane*. Additionally, a switch can change rules without notifying the *control plane*. Clearly, the issues related to TLS deployment represent a major challenge to security in the SDN context. A possible solution to such problems is the use of *middleboxes* to perform the communication between the *data plane* and the *control plane*. Sherwood et al. [89] propose a virtualization platform called FlowVisor that intermediates the communication between the network controller and the switches. Other examples of middleboxes used to enhance security are presented by Anwer et al. [69] and Fayazbakhsh et al. [71]. In particular, Kreutz et al. [64] present the vectors of the most common threats in SDN, such as DDoS attacks, and comment on the TLS challenges above.

Silva et al. [75] describe the use of an anti-eavesdropping technique based on multipath routing for SDN-based SCADA systems used in electrical smart grids. Centered on the confidentiality of the OpenFlow protocol, Kloti et al. [90] consider attacks that exploit flow aggregation to discover information about the network state and topology, which would not be visible otherwise. The authors use two modeling techniques, namely Microsoft's STRIDE and *attack trees*, to identify and explore threats to SDNs. Schlesinger et al. [79] analyze the problem of dividing the network in slices, thereby allowing traffic isolation, which can guarantee confidentiality in the communication.

Frequently, the mitigation mechanisms used should install firewall rules in the *data plane*. Porras et al. [66] propose a software extension to the NOX controller called FortNOX, which is intended to avoid conflicting rules to be installed in the data plane. FortNOX is also used as a security policy management tool. Also in the context of security policies, Naous et al. [73] propose the protocol *ident++* that allows the search for information or rules placed on hosts. Ballard et al. [77] propose the OPENSAFE and ALARMS languages to simplify the specification of security policies using the OpenFlow protocol. Jafarian et al. [76] present an elegant solution for the mitigation of malicious activities and protection of IP addresses against spoofing by enabling the network to randomly modify the IP addresses used.

Nonrepudiality and **AAA:** very few works have addressed exclusively these aspects. Some firewalls deal with these issues, but it is possible that SDN can ensure these properties using TLS in its communication. Further reading can be found in [91], which presents a survey discussing interesting aspects of security in SDN.

3.4. Performability

In recent years, the performability of the *control plane* has been a main concern. Despite the benefits of a centralized control logic in the SDN/OpenFlow architecture, Curtis et al. [54] point out that the current OpenFlow specification does not meet the demands of high performance networks. This is mainly due to the following reasons: (i) there is a high dependence on a central logic and on the global view of the network; (ii) it is possible that path latency can slow down the communication between the *control* and *data planes* during flow setup; and (iii) there is an excessive dependence on the *control plane*, demanding considerable resources to maintain this feature. To address these challenges,

[2] http://www.snort.org.

Table 4
SDN research efforts on performability.

SDN planes	QoS
Application plane	Wei et al. [93]
Control plane	Curtis et al. [54]
	Veisllari et al. [46]
	Egilmez et al. [94]
	Akella et al. [95]
	Huang et al. [96]
	Xiong et al. [97]
	Wang et al. [98]
	Machado et al. [99]
Data plane	Zhang et al. [92]
	Egilmez et al. [94]
	Wang et al. [98]
	Machado et al. [99]

the authors present DevoFlow, a modification of OpenFlow that reduces the amount of communication between the *control* and *data planes*, thereby reducing its overhead. DevoFlow achieves this goal by handling flows in the *data plane*. Additionally, Veisllari et al. [46] investigate the performability of the *control plane* with respect to scalability. Scalability is related to performance when we consider metrics such as delay, latency and throughput. The authors conclude that the current Internet flow definitions have high requirements on the processing rate of the SDN controller.

One of these requirements is the consistent population of flow tables regarding performance in traffic management. Zhang et al. [92] address the problem of redundant rules that can appear after successive insertions of flow rules. The authors discuss a compression method based on the combination of similar entries using techniques such as Huffman coding. Internet applications over the network have performance requirements with respect to the communication with the *control plane*. Efforts focusing on the performability of the Northbound API are presented by Wei et al. [93], who propose the use of caches to speed up the service of this interface. Table 4 categorizes the major research efforts related to performability in SDN, and next we discuss these efforts with respect to QoS (quality of service).

QoS: Sonkoly et al. [100] state that SDN developed slowly with respect to QoS support and that the current result is "even worse" than expected. In part this occurs due to several limitations of the devices present in the *data plane*, which is the same reason that makes the implementation of QoS difficult in traditional networks. The current version of the OpenFlow protocol supports only simple mechanisms to address QoS queues. Some counters provided by the *data plane*, *e.g.*, the number of packets received or flow duration, can be used to provide QoS on existing packets in the network, but are limited and inflexible. Additionally, the manipulation of existing switch queues is difficult because there is a lack of advanced interfaces to access their information. One solution is to use protocols such as *OF-config*, which offers a management interface with NETCONF features.

Egilmez et al. [94] propose an architecture for QoS called OpenQoS, which is used for grouping data streams and multimedia flows in sets of traffic classes allowing differential treatment for each type of class. Note that QoS-based approaches are inherently dependent on the queue concept and prioritization, which can be assisted by the programmability of the *control plane*. Egilmez et al. [101] address QoS for video streaming and deal with routing issues, such as packet loss. The authors consider large networks controlled by clusters of network controllers, and all the controllers decide jointly the best policy to enforce QoS in the network.

Other examples of QoS support in SDN are:

- *Cloud:* Akella et al. [95] address the problem of guaranteeing QoS requirements for cloud users, such as low delay. The most used QoS solution for multimedia applications is to differentiate the way in which different types of packets are forwarded. Also, the authors study bandwidth allocation for QoS using queuing techniques. The performance metrics used are response time and number of hops. Another effort towards QoS in clouds is presented by Huang et al. [96], which provide a theoretical and experimental analysis for end-to-end QoS provisioning.
- *Data stores:* Xiong et al. [97] advocate the use of SDN to manage the performance of distributed queries in data stores. The authors discuss how to control the priority of network traffic or make bandwidth reservations using queuing theory.
- *Servers:* Wang et al. [98] propose an autonomic QoS management mechanism for SDN by extending the OpenFlow and OF-Config protocols. The authors introduce a packet context-aware QoS model (PCaQoS) to provide self-configuration.
- *Policy Based Management:* Machado et al. [99] propose a policy refinement approach for QoS management. The authors propose a method capable of identifying QoS requirements and use PBM (Policy Based Management) mechanisms and natural language to translate these requirements into primitive switch configurations.

3.5. Traffic tolerance

Traffic tolerance enables the network to withstand unusual traffic profiles without compromising its expected behavior [7]. This discipline deals with traffic related questions, such as malicious attacks and legitimate traffic management. In the following, the most important studies in this topic are discussed. Table 5 summarizes the work presented in this section.

3.5.1. Legitimate traffic

Although protocols such as *SCTCP* and *MP-TCP* delegate traffic resilience to the network core, it is a somewhat limited approach because global network protection is difficult to achieve due to the lack of flexibility of traditional architectures. An alternative is to transfer this responsibility to the edge of the network, *i.e.*, to improve routing algorithms in order to make traffic management more resilient. Despite this seems more appealing and flexible, even a simple change to routing paths can lead to inappropriate solutions with respect to network performance and communication delay. Traditional, non-SDN network architectures do not offer support for advanced routing solutions, however, within an SDN architecture, software abstractions can be created to improve legitimate traffic resilience and complex routing schemes can be easily deployed.

Table 5
SDN research efforts on traffic tolerance.

SDN planes	Legitimate traffic	DDoS attacks
Application plane	N/A	N/A
Control plane	Taveira et al. [9]	Braga et al. [86]
	Rothenberg et al. [102]	Shin et al. [103]
	Agarwal et al. [5]	Michale et al. [104]
	Akyildiz et al. [105]	Giotis et al. [106]
	Taveira et al. [9]	Benton et al. [59]
	Ramos et al. [107]	Alcorn et al. [108]
	Benson et al. [109]	Belyaev et al. [110]
	Raza et al. [111]	Wang et al. [112]
	Venmani et al. [113]	Passito et al. [114]
	Qazi et al. [84]	Li et al. [115]
	Shimim et al. [116]	Shtern et al. [117]
	Silva et al. [118]	
	Laga et al. [119]	
	Xiaogang et al. [120]	
	Rodrigues et al. [121]	
	Bennesby et al. [122]	
Data plane	Taveira et al. [9]	Braga et al. [86]
	Ramos et al. [107]	Krishnan et al. [123]
	Bensom et al. [109]	Michale et al. [104]
	Venmani et al. [113]	Benton et al. [59]
	Sgambelluri et al. [124]	

A broad discussion on routing in SDN can be seen in the work of Rothenberg et al. [102]. According to the authors, the OpenFlow protocol represents a real opportunity for the deployment and evaluation of routing strategies. Typically, these solutions are implemented in the *control plane*, *i.e.*, the SDN controller is extended to deal with problems such as link failure, communication delay and load distribution. Agarwal et al. [5] deal with traffic engineering issues, but in the context of delay and packet losses. The work supports adaptive routing based on performance metrics, such as delay. The authors rely on the global network view to create a graph that represents all links available. After this, a mathematical formulation of the routing problem considering these metrics is produced, and a Fully Polynomial Time Approximation Scheme (FPTAS) is used to find the best solution for the problem. One advantage of this work is that it does not require protocol changes and can be used in scenarios where SDN is not completely deployed. Akyildiz et al. [105] deal with similar traffic issues. In general, the programmability offered by SDN encourages the use of classical algorithms, such as graph-based algorithms, to solve routing problems.

INFLEX is a framework that extends the network controller to create multiple routing planes that can be switched when a challenge is observed, for example, a link failure [9]. The core of the architecture lies on the differentiated services (DS) field of the IP protocol to create the notion of a routing plane for every packet in the network. Every host sets the DS field properly to guarantee that its packets will follow the same route. Additionally, hosts can request a new plane when a fault is observed (*e.g.*, if no response is received after a few seconds). Other authors, *e.g.*, Ramos et al. [107], similarly exploit the programmability of SDN to support alternative communication paths.

Benson et al. [109] address new traffic engineering strategies in data center networks to efficiently accommodate various types of traffic. Although the research area of traffic engineering is not focused exclusively on resilience, some of its concepts can be used to support resilience objectives, *e.g.*, prioritization of traffic profiles. In order to handle conflicting constraints, such as conflicting QoS requirements, the MeasuRouting framework presented by Raza et al. [111] can enforce QoS using traffic monitors that guarantee the demand of the network traffic.

With respect to link congestion, Venmani et al. [113] use OpenFlow to provide improvements to flow routing in backbone networks. These improvements include the use of the network controller to generate high-level policies to notify link failures. In this case, the controller can run a routine to recover the network back to its normal operation (*e.g.*, calculation of new paths and link failure detection). Sgambelluri et al. [124] remove this responsibility from the network controller and pass it to the *data plane*. The idea is to install backup rules with low priority, thus in case an error occurs the *data plane* itself can change the path used for communication. In scenarios where the network controller is usually overloaded, such solutions serve as an efficient alternative to avoid communication latency. To enforce capabilities related to policy management in the performance context, Qazi et al. [84] propose the use of middleboxes orchestrated by SIMPLE, a policy enforcement layer in the *control plane*. Additionally, the specification of packet-forwarding policies can be assisted by high-level languages, such as those proposed by Voellmy et al. [125] and Foster et al. [126].

Given that one of the main benefits of SDN is to allow flexibility in routing, several solutions for routing challenges have been investigated. Shimim et al. [116] and Silva et al. [118] deal primarily with multicast routing. They suggest the use of applications on the *control plane* to orchestrate flow behavior in order to provide multicast routing. Video routing is addressed by Laga et al. [119] and issues related to MPLS are discussed in the work of Xiaogang et al. [120]. Rodrigues et al. [121] discuss the optimization of network utilization across layers using bandwidth virtualization. The authors show that inefficiencies exist in links of data center WANs, but SDN can assist in addressing these restrictions. Bennesby et al. [122] address the problem of inter-domain routing with SDN support. The authors present performance and scalability results to demonstrate that SDN can help to mitigate the limitations of rigid BGP deployments, such as the difficulty in supporting architectural innovation. Additionally, an overview of transport networks in the context of SDN is provided by Alvizu et al. [127].

3.5.2. DDoS attacks

According to Mehdi et al. [87], the deployment of an anomaly detection system in the traditional network core is difficult mainly due to the low detection rate that these systems can provide with limited network information. In SDN, however, the *control plane* has a comprehensive view of the network, which facilitates the implementation of detection mechanisms. Mehdi et al. [87] present an overview of attack detection possibilities using SDN. The authors discuss four well-known algorithms: TRW-CB, MaxEnt, RateLimit and NETAD. Their study suggests that SDN is a platform suitable for the mitigation of DDoS attacks, mainly because of the use of standard protocols, services and interfaces, thus facilitating the deployment of new solutions.

Several strategies can be used to detect and mitigate DDoS attacks. Belyaev et al. [110] state that existing solutions to mitigate DDoS attacks can be classified as *active* (*e.g.*, based on the use of machine learning for detection) or *survival* (*e.g.*, trying to tolerate a DDoS attack). The latter is concerned with solutions based on load balancing. As pure load balancing is not effective during a DDoS attack, an iterative splitting of traffic paths where the network is overloaded may be necessary. This can increase the chances of tolerating a DDoS attack. For example, Wang et al. [112] present a mechanism to protect the control path against DDoS attacks by scaling the control channel capacity. This allows the network to handle a large number of flows, and makes the *control plane* more resilient.

Braga et al. [86] investigate the mitigation of DDoS attacks using Self-Organizing-Maps (SOM), a machine learning algorithm already used in traditional networks but with limited effects due to the restrictions of that architecture. Frequently, well-known solutions for traditional networks are implemented in the SDN context, as in the work of Ramadas et al. [157]. Further, Shin et al. [103] propose the insertion of triggers to control and change flow dynamics in the *data plane*. This can be used to expose the malicious flows for the detection and mitigation of DDoS attacks. For example, Krishnan et al. [123] present several detection methods and discuss which methods can be implemented in the *data plane*. Michale et al. [104] propose packet classification based on techniques such as prefix match and flow caches, to avoid the repeated classification of flows. Alcorn et al. [108] present a framework to model and simulate DoS and DDoS attacks in SDN/OpenFlow networks.

More recently, the use of information theory for packet classification has been investigated by Giotis et al. [106], who use entropy analysis for monitoring deviations in network behavior. Additionally, man-in-the-middle attacks are discussed by Benton et al. [59], who indicate that the adoption of TLS as a secure communication channel can present vulnerabilities. Passito et al. [114] present a solution that allows SDN domains to cooperate in the mitigation of DDoS attacks. Li et al. [115] use traffic engineering techniques to reduce the impact of a DDoS attack, and Shtern et al. [117] address the mitigation of low and slow distributed denial of service (LSDDoS), a variant of traditional DDoS that can compromise network applications by simulating their behavior.

3.6. Disruption tolerance

The distributed nature of network devices contributes to the unpredictability of delay in their communication. In addition, natural disasters, *e.g.*, hurricanes and earthquakes, often compromise links, preventing communication in the affected region and causing communication delays in other parts. Power outages and intermittent connection can also leave part of the network without operation. Issues related to these challenges, summarized in Table 6, comprise the disruption tolerance discipline. These are discussed in the following.

Connectivity: a threat constantly faced by networks is the disruption of the connectivity between its components. Link disruptions due to natural disasters or human interaction require the rapid establishment of alternative routes to restore in part the services affected. These new routes can generate bottlenecks in network devices that face an excessive demand for data processing. This might result in the delivery of degraded services to network users. Menth et al. [158] deal with link disruption issues and rerouting of packets in traditional networks, and illustrate how critical these questions are to the resilience of networks in general.

Despite the fact that SDN is appropriate for innovation of the network control tasks, the problems faced by traditional IP networks persist even when we consider the full potential of this new approach [132]. A centralized *control plane* with an overview of the network topology has instigated studies that focus on new solutions for legacy problems in distributed systems, such as link failure. However, as in SDN the responsibility for defining the communication paths between network components lies with the network controller, a new category of problems arises that are related to the availability of this component.

The controller is responsible for defining how packets will be forwarded at the *data plane*. Thus, the protection of the controller is the first critical point to address. Heller et al. [47] and Zhang et al. [92] deal with the *controller placement problem*. Their objective is to ensure there is connectivity between the controller and the switches. Beheshti et al. [51] also tackle this problem, but additionally deal with traffic control issues, such as link disruption and component failure. Hock et al. [49] present a framework with the same objectives, but also considering metrics such as latency, component failures and load balancing. Their major conclusion is the impossibility of finding an optimal solution to place the controller that satisfies various different criteria, *e.g.*, latency and backup communication. Note that this resilience discipline is strongly related to dependability.

Several studies focused on links and routing protection are available. Nguyen et al. [141] define algorithms for finding alternative paths if a link disruption occurs in the network; Stephens et al. [144] present an architecture called *Plinko*, which is provably resilient to t link failures when the size of flow tables in the *data plane* is sufficiently large to accommodate backup flow rules; Borokhovich et al. [147] use graph theory to model the network as a graph and run algorithms such as depth-first search (DFS) and breadth-first search (BFS) to analyze the routing problem and ensure connectivity when a failure occurs.

Delay: communication delays can occur due to the rupture of intermediate links and, to ensure connectivity, alternative communication paths can be used. Link delays resulting from congestion due to the intensive use of network resources may be difficult to conceal [159]. When the problem is less severe, routing algorithms may solve part of the problem, as shown by Heller et al. [47] and Hock et al. [49]. These efforts investigate the influence that the position of the controller has on the communication latency between controller and switch. Phemius et al. [138] point out that switch buffers in the *data plane* play an important role in the overall performance of the network.

Another possible solution is to exploit parallelism to avoid network delay, as proposed by Cai et al. [140]. Their system implements a middlebox that handles requests to the controller in parallel and uses alternative paths to communicate. The same principle can be used in the OFLOPS platform [143],

Table 6
SDN research efforts on disruption tolerance.

SDN Planes	Connectivity	Mobility	Delay	Energy
Application plane	N/A	Pupatwibul et al. [128] Namal et al. [129] Yuhong et al. [130] SchulzZander et al. [131]	N/A	N/A
Control plane	Yeganeh et al. [132] Heller et al. [47] Zhang et al. [92] Beheshti et al. [51] Hock et al. [49] Nguyen et al. [141] Stephens et al. [144] Borokhovich et al. [147]	Pupatwibul et al. [128] Sabbagh et al. [134] Namal et al. [129] Sahri et al. [137] Zhipu et al. [139] Guolin et al. [142] Jeon et al. [145] Yuhong et al. [130] Shengli et al. [148] Ding et al. [149] SchulzZander et al. [131] Lara et al. [150] Jagadeesan et al. [151] Sperotto et al. [152] Giust et al. [153] Kyoungjae et al. [154] Hyunsik et al. [155]	Yeganeh et al. [132] Heller et al. [47] Hock et al. [49] Phemius et al. [138] Cai et al. [140] Rotsos et al. [143] Mahmoodi et al. [146]	Heller et al. [133] Wang et al. [135] Pfeiffenberger et al. [136]
Data plane	N/A	Guolin et al. [142] Shengli et al. [148] Lara et al. [150] SeongMun et al. [156]	N/A	Heller et al. [133]

which offers support for the rapid development and test of network components. Further, in order to reduce the impact of communication delay on the performance of the network, Mahmoodi et al. [146] propose a modular redesign of the intermediate links between the core and mobile networks to handle increasing traffic volumes.

Mobility: in the context of wireless networks, mobility is an important characteristic for the convenience of users. Resilience strategies may use the concept of mobility in the face of a challenge, *e.g.*, a device may need to temporarily migrate to another region until normal service is re-established. Challenges related to user mobility are summarized in the work of Pupatwibul et al. [128], where the authors recognize that OpenFlow is a suitable technology for dealing with mobility issues. Sabbagh et al. [134] propose a solution based on reprogramming OpenFlow switches to solve the problem of relocation rules in order to maintain communication when the user moves from one point to another in network.

Namal et al. [129] present an architecture called OpenFlow Host Identity Protocol (OFHIP), which provides a mechanism for switches to change their IP addresses due to malicious attacks. Further, Sahri et al. [137] study failures of network components moving in the communication path. Despite the fact that these solutions are not exclusively focused on mobility, they use related concepts to mitigate failures. A few works address mobility in different contexts. Guolin et al. [142] define an architecture for heterogeneous radio access networks and deal with aspects of mobility, *e.g.*, QoS. Other examples are the works of Jeon et al. [145], Yuhong et al. [130], Shengli et al. [148] and Ding et al. [149], which advocate that SDN can be used for mobility. SchulzZander et al. [131] discuss the feasibility of Wi-Fi deployments in the SDN context. Lara et al. [150] propose *MobileFirst*, a clean-slate monitoring architecture that addresses concepts such as communication delay using routing based on VLAN tags. Further information about wireless and SDN can be found in the work of Jagadeesan et al. [151] and SeongMun et al. [156].

SDN offers an opportunity to facilitate the treatment of challenges in mobile networks, such as mobility management. Traditional solutions present limitations related to, for example, routing and continuity of active sessions. The use of decentralized device anchors represents an alternative to address these limitations. Further, the use of DMM (Distributed Mobility Management) in SDN has been advocated by Sperotto et al. [152] and in IETF proposals [153–155]. SDN can assist in mitigating these issues as it provides a flexible architecture for the deployment of network protocols and applications.

Energy: computer networks often demand scalable and massive services, which can give rise to challenges related to energy. Current data centers consume a large amount of energy and according to [133], energy issues in data centers are an important research topic. Briefly, the authors use techniques to dynamically adapt the power consumption in a data center network. One trend observed with respect to energy in the SDN context is that few studies are focused purely on energy issues. Some references are related to dependability when equipment failure is related to lack of power. These works were discussed in previous sections, since they are covered by other resilience disciplines.

Another study addressing energy aspects is presented by Wang et al. [135], which optimizes energy consumption by using different routing algorithms. The knowledge about the amount of energy spent by each device in the network to enforce QoS can also be used for energy saving purposes. Also, Pfeiffenberger et al. [136] advocate that SDN can be used to improve the management of energy aspects in communication networks.

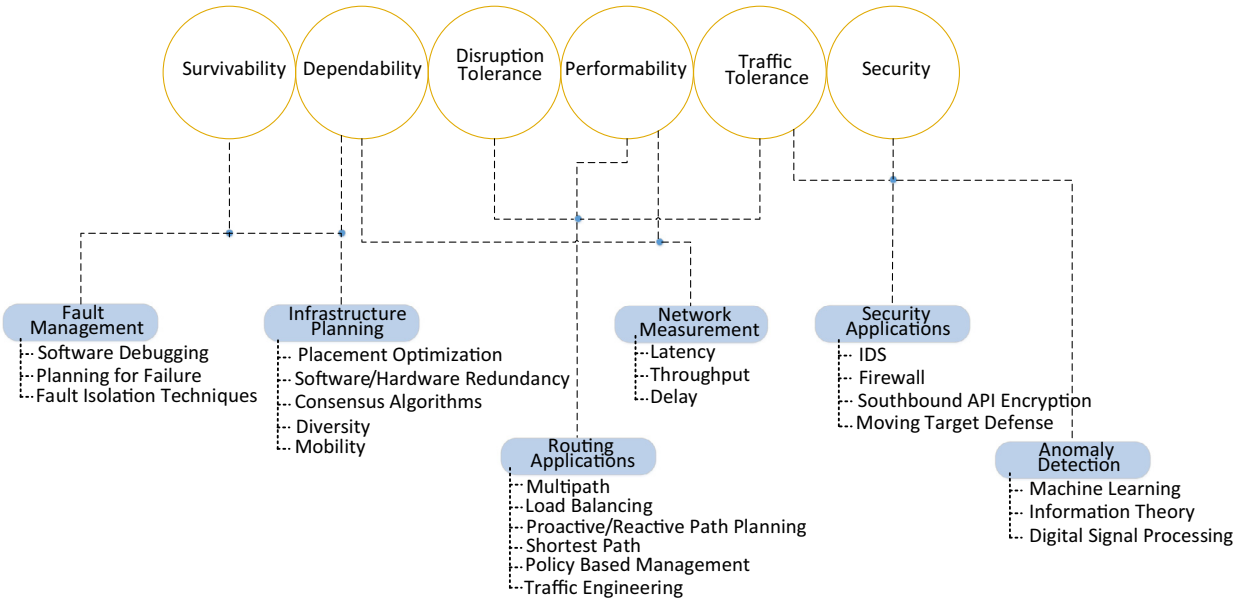

Fig. 3. Summary of resilience disciplines, major challenges and areas of interest, and concrete techniques.

The most studied topics about energy include optimization of energy consumption in the network and network resilience when power outages occur. The consequence of challenges associated with energy issues is link disruption. Thus, although energy issues are relevant in other disciplines discussed in this survey, these aspects are mostly related to ensuring connectivity among devices and disruption tolerance [7].

4. SDN resilience: solutions and challenges

The number of papers and research efforts that address different aspects of resilience in SDN is rapidly growing. This section discusses specific challenges and some of the major issues current research addresses. This is done with a focus on the identified areas and resilience disciplines that were introduced earlier. Subsequently, we present a summary of open research issues and areas that need further work.

4.1. Summary of challenges and current solutions

Fig. 3 summarizes the set of identified problems and challenges, according to the different areas and resilience disciplines. At the top of the diagram, the circles represent the six general resilience disciplines, and at the level below the major challenges and prominent research areas that are related to each resilience discipline are depicted. Under each of the challenges, examples of the various techniques, approaches, and concrete instantiations indicate the specific research focus related to the different challenges. They represent the issues most commonly addressed within the discussed literature.

The modular architecture of SDN enables more flexible ways to manage traffic flows [10]. Consequently, resilience solutions based on ***Routing Applications*** are employed in several disciplines (*e.g.*, they are being used in the context of performability, traffic tolerance and disruption tolerance). For instance, backup paths and multipath routing can be used to protect the communication network from link disruption and energy outages [136]. Performability also relies on traffic engineering techniques and load balancing to enforce QoS requirements [96]. Further, Policy Based Management can be used to add flexibility to these solutions [99]. Such schemes are usually implemented through extensions of the *control plane* with applications that can monitor and manage traffic via the OpenFlow protocol. Software abstractions, such as topology graphs, can be used to simplify the management of traffic flows and routing [19], thus enabling the use of shortest paths and minimum spanning trees to find optimal solutions for traffic routing.

In the context of ***Infrastructure Planning*** the controller placement problem plays an important role. Since it shares similarity with the classic *facility location* optimization problem [160], several proposed solutions use a graph representations of the network topology to determine the optimal placement of network controllers [47,49]. Also, solutions based on hardware and software redundancy have been widely investigated [35]. This is due to the fact that SDN offers a flexible architecture to manage redundant devices [30]. Further, schemes such as consensus algorithms to elect a new replica in case of a failure help to maintain consistency between components and their replicas [161]. In wireless networks, research indicates that SDN can assist with the implementation of solutions to guarantee connectivity between devices, *e.g.*, through software abstractions to change IP addresses of devices that migrate and re-route flows to guarantee communication [76].

In the investigated papers ***Fault Management*** often exploits the programmability features offered by the *control plane*, which enables the implementation of network applications related to software debugging [33]. Also, there are sophisticated monitoring applications capable of

Table 7
Key findings observed about the current research on resilience in SDN.

Discipline	Key aspects observed
Survivability	(i) There are cost issues that prevent the deployment of fully fault tolerant systems; (ii) management requirements of redundant devices can be high, for example, to maintain their consistency.
Dependability	(i) Controller placement is the most studied problem in this area; (ii) maintainability is an uncovered field that can give rise to interesting research, for example, with the addition of a dedicated *management plane* to the SDN architecture.
Security	(i) Several works focus on porting solutions used in traditional networks (*e.g.*, firewalls, IDSes) to SDN. Their main goal is to implement these techniques with more flexibility; (ii) mis-configurations and human-dependence in the use of TLS between the controller and switches can compromise the integrity and confidentiality of the communication.
Performability	(i) Although QoS support in SDN is far from optimal, several solutions are more flexible than existing ones in traditional networks; (ii) the controller placement is an issue that can impact communication delay, latency and throughput of the network.
Traffic Tolerance	(i) This is the most developed resilience discipline because the SDN architecture has been traditionally concerned with the innovation of routing protocols; (ii) well-known solutions to mitigate DDoS attacks can be successfully implemented in SDN.
Disruption Tolerance	(i) Again, controller placement is critical for protecting the network from disruption; (ii) solutions related to survivability are frequently used, such as path redundancy and link redundancy.

collecting information about topology changes, device crashes and link disruptions. These applications might also perform fault isolation, thereby creating a reliable environment for fault detection and mitigation. The use of *group tables* [162] in an OpenFlow switch is an example of how SDN can simplify capacity planning, by enabling the definition of backup flow rules in the switch.

Resilience approaches that rely upon ***Network Measurement*** are the most effective when they focus on measuring latency, throughput and delay. These metrics can be used for assessing QoS and the degree of dependability that the network can offer.

The work on ***Security Applications*** can be divided into two broad sets: *(i)* security solutions built on top of SDN and *(ii)* security of the SDN architecture itself. Within the first group, there are several implementations of firewalls and IDSes that can perform their functions more flexibly and with lower management cost [70]. Within the second group, research is focused on mitigating intrusions in the *control plane*; vulnerabilities in the TLS protocol; and protecting the *control plane* against malicious attacks using network-wide policies and moving target defense [74].

Finally, there is a large amount of work on ***Anomaly Detection***, which has a strong relationship with the security discipline, but also plays an important role in the traffic tolerance discipline. Anomaly detection schemes specific to SDN rely on a global network view to collect flow statistics and perform packet sampling. Most of these solutions rely on machine learning, information theory and digital signal processing techniques [163].

Ultimately, the resilience challenges observed can be divided into two classes. The first class refers to challenges related to the SDN architecture itself independently of any given implementation (*e.g.*, related to infrastructure planning and network measurement). For example, the controller placement is a theoretical problem that is relevant regardless of the controller implementation. The second class subsumes resilience challenges that depend on specific SDN implementations (*e.g.*, routing and security applications). In this case, the solutions in the literature are frequently based on the OpenFlow specification or highly dependent on the functionality provided by the controller implementation.

4.2. Open research questions and lessons learned

Several research questions related to resilience in SDN remain open. For example, high hardware costs related to fully fault tolerant systems can be partially mitigated in this scenario through the use of virtualized infrastructures. In this context, Network Functions Virtualization (NFV) [164] in the *application plane* can assist the development of new solutions in this area. There is also no real resilience metrics related to software implementations and their quality. Thus, Software Engineering practices could be a source for such new metrics to ensure a methodology for software implementations. Additionally, emerging types of traffic profiles suggest that applications related to traffic classification will be very useful for future SDN environments. This should be an active area of research for the next years, as well as the development of monitoring applications that rely on the global network view supported by the SDN architecture. Finally, the initial proposal of the OpenFlow protocol supports limited QoS capabilities, but the development of new protocols can create novel ways to tag flows, enabling sophisticated applications to enforce QoS in the network. Table 7 highlights these points and summarizes the main lessons learned from this study. Note that in addition to specific issues related to these individual disciplines, there is also the need to address resilience challenges that span across several disciplines. This might require the co-ordination of resilience mechanisms that operate at different layers and systems elements (*i.e.*, multi-level resilience), correlating data and also taking coordinated actions. Ultimately, an overall resilience architecture would then be able to discover network and systems anomalies more quickly, and enforce countermeasures at the most appropriate locations.

5. Concluding remarks

Resilience in software-defined networking (SDN) is the subject of intense investigation by the academic and industrial community in general. As SDN is a relatively new concept, a wide range of solutions to classical network problems have been revisited using this architecture, and many problems continue to be challenging. In this article, we have presented a comprehensive survey on the support for

resilience in the SDN architecture, categorizing the existing research efforts on resilience across the different SDN conceptual planes. Furthermore, this survey has presented an overall view of resilience research, describing the trends and key aspects observed, as well as the evolution of this area since 2008, when the widely-adopted OpenFlow protocol was proposed.

The number of research projects that address resilience aspects in SDN has grown significantly. This can be observed by the number of papers included in our survey, and also by the number of research calls issued in this topic recently. The main result of our survey is a comprehensive view of the research space in SDN resilience demonstrating that *(i)* the *data plane* can be protected against link disruption, device failures and malicious attacks using applications placed in the *control* or *application planes*; *(ii)* the *control plane* has resilience requirements related to the consistency between several network controller instances, the security of these devices and general fault management over the entire network. There are several ways to decide where network controllers will be placed and this decision is critical for network operation. Additional controllers may be deployed according to security and survivability requirements; *(iii)* the *application plane* can accommodate several types of network applications, thus promoting research on more sophisticated resilience systems to protect the network against a wide range of challenges. High-level policy languages, such as Procera [125] and Pyretic [126], and troubleshooting systems can also be used to facilitate these tasks.

We emphasize that many of the resilience challenges are due to limitations in the implementation of the components used to realize the SDN paradigm. For example, *(i)* the OpenFlow protocol can be unsafe if TLS is not set up correctly; *(ii)* the Floodlight controller exposes almost all of its functionality through a REST API (possibly allowing illegitimate applications to gather network data); and *(iii)* the *listener mode* functionality (present in many OpenFlow switches) may allow the establishment of connections in a pre-configured port without authentication. Despite the efforts reported in this survey, there are still a number of open issues related to the resilience disciplines investigated, such as the co-ordination of different types of resilience schemes regarding performance and consistency. Consequently, research that takes a more systematic view of resilience systems is required (*e.g.*, considering resilience aspects across different system layers). This article assists in the identification of these aspects that demand further research.

References

[1] B. Fortz, M. Thorup, Optimizing OSPF/IS-IS weights in a changing world, IEEE J. Select. Areas Commun. 20 (4) (2006) 756–767, doi:10.1109/JSAC.2002.1003042.

[2] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: past, present, and future of programmable networks, IEEE Commun. Surv. Tutorials 16 (3) (2014) 1617–1634, doi:10.1109/SURV.2014.012214.00180.

[3] D. Klein, M. Jarschel, An OpenFlow extension for the OMNeT++ INET framework, in: International Conference on Simulation Tools and Techniques (ICST), Brussels, Belgium, 2013, pp. 322–329.

[4] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, J. van der Merwe, The case for separating routing from routers, in: Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture, in: FDNA'04, ACM, New York, NY, USA, 2004, pp. 5–12, doi:10.1145/1016707.1016709.

[5] S. Agarwal, M. Kodialam, T. Lakshman, Traffic engineering in software defined networks, in: IEEE Proceedings of the INFOCOM, 2013, pp. 2211–2219, doi:10.1109/INFCOM.2013.6567024.Turin

[6] E. Cetinkaya, J. Sterbenz, A taxonomy of network challenges, in: 9th International Conference on the Design of Reliable Communication Networks (DRCN), 2013, 2013, pp. 322–330.

[7] J.P.G. Sterbenz, D. Hutchison, E.K. Cetinkaya, A. Jabbar, J.P. Rohrer, M. Scholler, P. Smith, Resilience and survivability in communication networks: strategies, principles, and survey of disciplines, Comput. Netw 54 (8) (2010) 1245–1265, doi:10.1016/j.comnet.2010.03.005.

[8] A. Schaeffer-Filho, P. Smith, A. Mauthe, D. Hutchison, Network resilience with reusable management patterns, IEEE Commun. Mag. 52 (7) (2014) 105–115, doi:10.1109/MCOM.2014.6852091.

[9] C.R.G. Taveira João A. Landa R., P. George, Software-defined network support for transport resilience, in: IEEE Network Operations and Management Symposium (NOMS), Krakow, 2014, pp. 1–8, doi:10.1109/NOMS.2014.6838243.

[10] N. Feamster, J. Rexford, E. Zegura, The road to SDN, Queue, vol. 11, ACM, 2013, pp. 20:20–20:40, doi:10.1145/2559899.2560327.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74, doi:10.1145/1355734.1355746.

[12] Y. Jarraya, T. Madi, M. Debbabi, A survey and a layered taxonomy of software-defined networking, IEEE Commun. Surv. Tutorials 16 (4) (2014) 1955–1980, doi:10.1109/COMST.2014.2320094.

[13] H. Cui, Y. Zhu, Y. Yao, L. Yufeng, Y. Liu, Design of intelligent capabilities in SDN, in: 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014, pp. 1–5, doi:10.1109/VITAE.2014.6934459.

[14] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, ONOS: towards an open, distributed SDN OS, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, ACM, New York, NY, USA, 2014, pp. 1–6, doi:10.1145/2620728.2620744.

[15] Open Networking Foundation, OpenFlow—Version 1.0.0 (Wire Protocol 0x01), 2009, https://www.opennetworking.org/sdn-resources/onf-specifications/openflow (accessed August 2014).

[16] H. Farhadi, P. Du, A. Nakao, Enhancing OpenFlow actions to offload packet-in processing, Asia-Pacific Network Operations and Management Symposium (APNOMS) (2014) 1–6, doi:10.1109/APNOMS.2014.6996603.

[17] J. de Almeida Amazonas, G. Santos-Boada, J. Solé-Pareta, A critical review of OpenFlow/SDN-based networks, in: 16th International Conference on Transparent Optical Networks (ICTON), 2014, pp. 1–5, doi:10.1109/ICTON.2014.6876509.

[18] H. Nakayama, T. Mori, S. Ueno, Y. Watanabe, T. Hayashi, An implementation model and solutions for stepwise introduction of SDN, 16th Asia-Pacific Network Operations and Management Symposium (APNOMS) (2014) 1–4, doi:10.1109/APNOMS.2014.6996576.

[19] G. Pantuza, F. Sampaio, L.F. Vieira, D. Guedes, M.A. Vieira, Network management through graphs in software defined networks, in: 10th International Conference on Network and Service Management (CNSM), 2014, pp. 400–405, doi:10.1109/CNSM.2014.7014202.

[20] A. Greenberg, J. Hamilton, D.A. Maltz, P. Patel, The cost of a cloud: research problems in data center networks, SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73, doi:10.1145/1496091.1496103.

[21] A. Lara, A. Kolasani, B. Ramamurthy, Network innovation using OpenFlow: a survey, IEEE Commun. Surv. Tutorials 16 (1) (2014) 493–512, doi:10.1109/SURV.2013.081313.00105.

[22] L. Schiff, M. Borokhovich, S. Schmid, Reclaiming the brain: useful OpenFlow functions in the data plane, in: Proceedings of the 13th ACM Workshop on Hot Topics in Networks, in: HotNets-XIII, ACM, New York, NY, USA, 2014, pp. 7:1–7:7, doi:10.1145/2670518.2673874.

[23] M. Casado, T. Koponen, S. Shenker, A. Tootoonchian, Fabric: a retrospective on evolving SDN, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, in: HotSDN'12, ACM, New York, NY, USA, 2012, pp. 85–90, doi:10.1145/2342441.2342459.

[24] V. Caraguay, A. Leonardo, L.I. Barona Lopez, L.J. Garcia Villalba, Evolution and challenges of software defined networking, in: IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–7.

[25] F. Hu, Q. Hao, K. Bao, A survey on software defined networking (SDN) and OpenFlow: from concept to implementation, IEEE Commun. Surv. Tutorials 16 (2014) 2181–2206, doi:10.1109/COMST.2014.2326417.
[26] A. Kwasinski, W. Weaver, P. Chapman, P. Krein, Telecommunications power plant damage assessment caused by Hurricane Katrina—site survey and follow-up results, in: 28th Annual International Telecommunications Energy Conference (INTELEC'06), 2006, pp. 1–8, doi:10.1109/INTLEC.2006.251644.
[27] I. Onyeji, M. Bazilian, C. Bronk, Cyber security and critical energy infrastructure, Electric. J. 27 (2) (2014) 52–60.
[28] P. Cholda, A. Mykkeltveit, B. Helvik, O. Wittner, A. Jajszczyk, A survey of resilience differentiation frameworks in communication networks, IEEE Commun. Surv. Tutorials 9 (4) (2007) 32–55, doi:10.1109/COMST.2007.4444749.
[29] P. Smith, D. Hutchison, J. Sterbenz, M. Schöller, A. Fessi, M. Karaliopoulos, C. Lac, B. Plattner, Network resilience: a systematic approach, IEEE Commun. Mag. 49 (7) (2011) 88–97, doi:10.1109/MCOM.2011.5936160.
[30] M. Reitblatt, M. Canini, A. Guha, N. Foster, FatTire: declarative fault tolerance for software-defined networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 109–114, doi:10.1145/2491185.2491187.
[31] B. Chandrasekaran, T. Benson, Tolerating SDN application failures with LegoSDN, in: Proceedings of the 13th ACM Workshop on Hot Topics in Networks, in: HotNets-XIII, ACM, New York, NY, USA, 2014, pp. 22:1–22:7, doi:10.1145/2670518.2673880.
[32] B. Heller, C. Scott, N. McKeown, S. Shenker, A. Wundsam, H. Zeng, S. Whitlock, V. Jeyakumar, N. Handigol, J. McCauley, K. Zarifis, P. Kazemian, Leveraging SDN layering to systematically troubleshoot networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 37–42, doi:10.1145/2491185.2491197.
[33] M. Canini, D. Venzano, P. Perešíni, D. Kostić, J. Rexford, A NICE way to test OpenFlow applications, in: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, NSDI'12, USENIX Association, Berkeley, CA, USA, 2012, p. 10.
[34] C. Scott, A. Wundsam, B. Raghavan, A. Panda, A. Or, J. Lai, E. Huang, Z. Liu, A. El-Hassany, S. Whitlock, H. Acharya, K. Zarifis, S. Shenker, Troubleshooting backbox SDN control software with minimal causal sequences, in: Proceedings of the 2014 ACM Conference on SIGCOMM, in: SIGCOMM'14, ACM, New York, NY, USA, 2014, pp. 395–406, doi:10.1145/2619239.2626304.
[35] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, A. Vahdat, B4: experience with a globally-deployed software defined wan, SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 3–14, doi:10.1145/2534169.2486019.
[36] D. Williams, H. Jamjoom, Cementing high availability in OpenFlow with RuleBricks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 139–144, doi:10.1145/2491185.2491206.
[37] F. Botelho, F. Valente Ramos, D. Kreutz, A. Bessani, On the feasibility of a consistent and fault-tolerant data store for SDNs, in: Second European Workshop on Software Defined Networks (EWSDN), 2013, pp. 38–43, doi:10.1109/EWSDN.2013.13.
[38] H.H. Liu, S. Kandula, R. Mahajan, M. Zhang, D. Gelernter, Traffic engineering with forward fault correction, in: Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM'14, ACM, New York, NY, USA, 2014, pp. 527–538, doi:10.1145/2619239.2626314.
[39] J. Li, J. Hyun, J.-H. Yoo, S. Baik, J.-K. Hong, Scalable failover method for data center networks using OpenFlow, in: IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–6, doi:10.1109/NOMS.2014.6838393.
[40] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, P. Skoldstrom, Scalable fault management for OpenFlow, in: IEEE International Conference on Communications (ICC), 2012, pp. 6606–6610, doi:10.1109/ICC.2012.6364688.
[41] P. Fonseca, R. Bennesby, E. Mota, A. Passito, A replication component for resilient OpenFlow-based networking, in: IEEE Network Operations and Management Symposium (NOMS), 2012, pp. 933–939, doi:10.1109/NOMS.2012.6212011.
[42] A. Tootoonchian, Y. Ganjali, HyperFlow: a distributed control plane for OpenFlow, in: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10, USENIX Association, Berkeley, CA, USA, 2010, p. 3.
[43] L. Muller, R. Oliveira, M. Luizelli, L. Gaspary, M. Barcellos, Survivor: an enhanced controller placement strategy for improving SDN survivability, in: IEEE Global Communications Conference (GLOBECOM), 2014, pp. 1909–1915, doi:10.1109/GLOCOM.2014.7037087.
[44] Y. Zhang, N. Beheshti, R. Manghirmalani, NetRevert: rollback recovery in SDN, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14, ACM, New York, NY, USA, 2014, pp. 231–232, doi:10.1145/2620728.2620779.
[45] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. Dependable Secure Comput. 1 (1) (2004) 11–33, doi:10.1109/TDSC.2004.2.
[46] R. Veisllari, N. Stol, S. Bjornstad, C. Raffaelli, Scalability analysis of SDN-controlled optical ring MAN with hybrid traffic, in: IEEE International Conference on Communications (ICC), 2014, pp. 3283–3288, doi:10.1109/ICC.2014.6883827.
[47] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN'12, ACM, New York, NY, USA, 2012, pp. 7–12, doi:10.1145/2342441.2342444.
[48] D. Li, L. Ruan, L. Xiao, M. Zhu, W. Duan, Y. Zhou, M. Chen, Y. Xia, M. Zhu, High availability for Non-stop network controller, in: IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014, pp. 1–5, doi:10.1109/WoWMoM.2014.6918986.
[49] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, P. Tran-Gia, Pareto-optimal resilient controller placement in SDN-based core networks, in: 25th International Teletraffic Congress (ITC), 2013, pp. 1–9.
[50] F.J. Ros, P.M. Ruiz, Five Nines of Southbound Reliability in Software-defined Networks, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, ACM, New York, NY, USA, 2014, pp. 31–36, doi:10.1145/2620728.2620752.
[51] N. Beheshti, Y. Zhang, Fast failover for control traffic in Software-defined Networks, in: IEEE Global Communications Conference (GLOBECOM), 2012, pp. 2665–2670, doi:10.1109/GLOCOM.2012.6503519.
[52] M. Santos, B. Nunes, K. Obraczka, T. Turletti, B. de Oliveira, C. Margi, Decentralizing SDN's control plane, in: IEEE 39th Conference on Local Computer Networks (LCN), 2014, pp. 402–405, doi:10.1109/LCsN.2014.6925802.
[53] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, R. Kompella, Towards an elastic distributed SDN controller, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 7–12, doi:10.1145/2491185.2491193.
[54] A.R. Curtis, J.C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, S. Banerjee, DevoFlow: scaling flow management for high-performance networks, in: Proceedings of the ACM SIGCOMM 2011 Conference, SIGCOMM'11, ACM, New York, NY, USA, 2011, pp. 254–265.
[55] Y.-J. Chen, F.-Y. Lin, L.-C. Wang, B.-S. Lin, A dynamic security traversal mechanism for providing deterministic delay guarantee in SDN, in: IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014, pp. 1–6, doi:10.1109/WoWMoM.2014.6918983.
[56] X. Wang, Z. Liu, J. Li, B. Yang, Y. Qi, Tualatin: Towards network security service provision in cloud datacenters, in: 23rd International Conference on Computer Communication and Networks (ICCCN), 2014, pp. 1–8, doi:10.1109/ICCCN.2014.6911782.
[57] S. Seeber, G.D. Rodosek, Improving network security through SDN in cloud scenarios, in: 10th International Conference on Network and Service Management (CNSM), 2014, pp. 376–381, doi:10.1109/CNSM.2014.7014198.
[58] M. Tasch, R. Khondoker, R. Marx, K. Bayarou, Security analysis of security applications for software defined networks, in: Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference, in: AINTEC'14, ACM, New York, NY, USA, 2014, pp. 23:23–23:30, doi:10.1145/2684793.2684797.
[59] K. Benton, L.J. Camp, C. Small, OpenFlow vulnerability assessment, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 151–152, doi:10.1145/2491185.2491222.
[60] H. Li, P. Li, S. Guo, S. Yu, Byzantine-resilient secure software-defined networks with multiple controllers, in: IEEE International Conference on Communications (ICC), 2014, pp. 695–700, doi:10.1109/ICC.2014.6883400.
[61] A. Zaalouk, R. Khondoker, R. Marx, K. Bayarou, OrchSec: an orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions, in: IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–9, doi:10.1109/NOMS.2014.6838409.
[62] A.K. Nayak, A. Reimers, N. Feamster, R. Clark, Resonance: dynamic access control for enterprise networks, in: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, WREN'09, ACM, New York, NY, USA, 2009, pp. 11–18, doi:10.1145/1592681.1592684.

[63] L. Schehlmann, S. Abt, H. Baier, Blessing or curse? Revisiting security aspects of software-defined networking, in: 10th International Conference on Network and Service Management (CNSM), 2014, pp. 382–387, doi:10.1109/CNSM.2014.7014199.

[64] D. Kreutz, F.M. Ramos, P. Verissimo, Towards Secure and Dependable Software-defined Networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 55–60, doi:10.1145/2491185.2491199.

[65] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, G. Jiang, NetFuse: short-circuiting traffic surges in the cloud, in: IEEE International Conference on Communications (ICC), 2013, pp. 3514–3518, doi:10.1109/ICC.2013.6655095.

[66] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, G. Gu, A security enforcement kernel for OpenFlow networks, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN'12, ACM, New York, NY, USA, 2012, pp. 121–126, doi:10.1145/2342441.2342466.

[67] D. Mazières, D. Shasha, Building secure file systems out of byzantine storage, in: Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing, PODC'02, ACM, New York, NY, USA, 2002, pp. 108–117, doi:10.1145/571825.571840.

[68] J. Collings, J. Liu, An OpenFlow-based prototype of SDN-oriented stateful hardware firewalls, in: IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 525–528, doi:10.1109/ICNP.2014.83.

[69] B. Anwer, T. Benson, N. Feamster, D. Levin, J. Rexford, A slick control plane for network middleboxes, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, in: HotSDN'13, ACM, New York, NY, USA, 2013, pp. 147–148, doi:10.1145/2491185.2491223.

[70] H. Hu, W. Han, G.-J. Ahn, Z. Zhao, FLOWGUARD: building robust firewalls for software-defined networks, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, ACM, New York, NY, USA, 2014, pp. 97–102, doi:10.1145/2620728.2620749.

[71] S.K. Fayazbakhsh, V. Sekar, M. Yu, J.C. Mogul, FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN'13, ACM, New York, NY, USA, 2013, pp. 19–24, doi:10.1145/2491185.2491203.

[72] T. Xing, D. Huang, L. Xu, C.-J. Chung, P. Khatkar, SnortFlow: a OpenFlow-based intrusion prevention system in cloud environment, in: Second GENI Research and Educational Experiment Workshop (GREE), 2013, pp. 89–92, doi:10.1109/GREE.2013.25.

[73] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, N. Zeldovich, Delegating network security with more information, in: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, WREN'09, ACM, New York, NY, USA, 2009, pp. 19–26, doi:10.1145/1592681.1592685.

[74] P. Kampanakis, H. Perros, T. Beyene, SDN-based solutions for moving target defense network protection, in: IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014, pp. 1–6, doi:10.1109/WoWMoM.2014.6918979.

[75] E.G. Silva, L. Knob, J.A. Wickboldt, L.P. Gaspary, L.Z. Granville, A. Schaeffer-Filho, Capitalizing on SDN-based SCADA systems: an anti-eavesdropping case-study, in: 15th IFIP/IEEE Symposium on Integrated Network and Service Management (IM'2015), 2015, pp. 165–173.Ottawa, Canada

[76] J.H. Jafarian, E. Al-Shaer, Q. Duan, OpenFlow random host mutation: transparent moving target defense using software defined networking, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN'12, ACM, New York, NY, USA, 2012, pp. 127–132, doi:10.1145/2342441.2342467.

[77] J.R. Ballard, I. Rae, A. Akella, Extensible and scalable network monitoring using OpenSAFE, in: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10, USENIX Association, Berkeley, CA, USA, 2010, p. 8.

[78] R. Smeliansky, SDN for network security, in: First International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014, pp. 1–5, doi:10.1109/MoNeTeC.2014.6995602.

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

[80] Z. Abaid, M. Rezvani, S. Jha, MalwareMonitor: an SDN-based framework for securing large networks, in: Proceedings of the 2014 CoNEXT on student workshop, CoNEXT Student Workshop'14, ACM, New York, NY, USA, 2014, pp. 40–42, doi:10.1145/2680821.2680829.

[81] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, M. Tyson, FRESCO: modular composable security services for software-defined networks, in: Network and Distributed System Security Symposium (NDSS), 2013, pp. 1–16.

[82] S. Kumar, T. Kumar, G. Singh, M.S. Nehra, OpenFlow switch with intrusion detection system, Int. J. Sci. Res. Eng. Techonol. (IJSRET) 1 (2012) 1–4.

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

[84] Z.A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, SIMPLE-flying middlebox policy enforcement using SDN, in: Proceedings of the ACM SIGCOMM 2013, New York, NY, USA, 2013, pp. 27–38, doi:10.1145/2486001.2486022.

[85] S. Son, S. Shin, V. Yegneswaran, P. Porras, G. Gu, Model checking invariant security properties in OpenFlow, in: IEEE International Conference on Communications (ICC), IEEE, 2013, pp. 1974–1979.

[86] R. Braga Braga, E. Mota Mota, A. Passito Passito, Lightweight DDoS flooding attack detection using NOX/OpenFlow, in: Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks, LCN'10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 408–415, doi:10.1109/LCN.2010.5735752.

[87] S.A. Mehdi, J. Khalid, S.A. Khayam, Revisiting traffic anomaly detection using software defined networking, in: Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, RAID'11, Springer-Verlag, Berlin/Heidelberg, 2011, pp. 161–180.

[88] K. Giotis, G. Androulidakis, V. Maglaris, Leveraging SDN for efficient anomaly detection and mitigation on legacy networks, in: Third European Workshop on Software Defined Networks (EWSDN), 2014, pp. 85–90, doi:10.1109/EWSDN.2014.24.

[89] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, Flowvisor: a network virtualization layer, OpenFlow Switch Consortium, Tech. Rep. (2009).

[90] R. Kloti, OpenFlow: A security analysis, in: IEEE Proceedings of the Workshop on Secure Network Protocols (NPSec), 2013, pp. 1–6.

[91] S. Scott-Hayward, G. O'Callaghan, S. Sezer, SDN security: a survey, in: IEEE SDN for Future Networks and Services (SDN4FNS), IEEE, 2013, pp. 1–7.

[92] Y. Zhang, S. Natarajan, X. Huang, N. Beheshti, R. Manghirmalani, A compressive method for maintaining forwarding states in SDN Controller, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, ACM, New York, NY, USA, 2014, pp. 139–144, doi:10.1145/2620728.2620759.

[93] W. Zhou, L. Li, W. Chou, SDN Northbound REST API with Efficient Caches, in: IEEE International Conference on Web Services (ICWS), 2014, pp. 257–264, doi:10.1109/ICWS.2014.46.

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

[95] A. Akella, K. Xiong, Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN), in: IEEE 12th International Conference on Dependable, Autonomic and Secure Computing (DASC), 2014, pp. 7–13, doi:10.1109/DASC.2014.11.

[96] J. Huang, Q. Duan, Q. Chen, Y. Sun, Y. Tanaka, W. Wang, Guaranteeing end-to-end quality-of-service with a generic routing approach, ACM SIGAPP Appl. Comput. Rev. 14 (2) (2014) 8–22, doi:10.1145/2656864.2656865.

[97] P. Xiong, H. Hacigumus, J.F. Naughton, A software-defined networking based approach for performance management of analytical queries on distributed data stores, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD'14, ACM, New York, NY, USA, 2014, pp. 955–966, doi:10.1145/2588555.2593681.

[98] W. Wendong, Q. Qinglei, G. Xiangyang, H. Yannan, Q. Xirong, Autonomic QoS management mechanism in software defined network, China Commun. 11 (7) (2014) 13–23, doi:10.1109/CC.2014.6895381.

[99] C. Cleder Machado, L. Zambenedetti Granville, A. Schaeffer-Filho, J. Araujo Wickboldt, Towards SLA policy refinement for QoS management in software-defined networking, in: IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014, pp. 397–404, doi:10.1109/AINA.2014.148.

[100] B. Sonkoly, A. Gulyas, F. Nemeth, J. Czentye, K. Kurucz, B. Novak, G. Vaszkun, On QoS support to Ofelia and OpenFlow, in: European Workshop on Software Defined Networking (EWSDN), 2012, pp. 109–113, doi:10.1109/EWSDN.2012.26.

[101] H. Egilmez, S. Civanlar, A. Tekalp, A distributed QoS routing architecture for scalable video streaming over multi-domain OpenFlow networks, in: 19th IEEE International Conference on Image Processing (ICIP), 2012, pp. 2237–2240, doi:10.1109/ICIP.2012.6467340.

[102] C.E. Rothenberg, M.R. Nascimento, M.R. Salvador, C.N.A. Corrêa, S. Cunha de Lucena, R. Raszuk, Revisiting routing control platforms with the eyes and muscles of software-defined networking, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN'12, ACM, New York, NY, USA, 2012, pp. 13–18, doi:10.1145/2342441.2342445.

[103] S. Shin, V. Yegneswaran, P. Porras, G. Gu, AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS'13, ACM, New York, NY, USA, 2013, pp. 413–424, doi:10.1145/2508859.2516684.

[104] L. Mchale, J. Case, P. Gratz, A. Sprintson, Stochastic pre-classification for SDN data plane matching, in: IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 596–602, doi:10.1109/ICNP.2014.95.

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

[106] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Comput. Netw. 62 (0) (2014) 122–136.

[107] R. Ramos, M. Martinello, C. Esteve Rothenberg, SlickFlow: resilient source routing in data center networks unlocked by OpenFlow, in: IEEE 38th Conference on Local Computer Networks (LCN), 2013, pp. 606–613.

[108] J. Alcorn, C. Chow, A framework for large-scale modeling and simulation of attacks on an OpenFlow network, in: 23rd International Conference on Computer Communication and Networks (ICCCN), 2014, pp. 1–6, doi:10.1109/ICCCN.2014.6911848.

[109] T. Benson, A. Anand, A. Akella, M. Zhang, MicroTE: fine grained traffic engineering for data centers, in: Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies, CoNEXT '11, ACM, New York, NY, USA, 2011, pp. 8:1–8:12, doi:10.1145/2079296.2079304.

[110] M. Belyaev, S. Gaivoronski, Towards load balancing in SDN-networks during DDoS-attacks, in: First International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014, pp. 1–6, doi:10.1109/MoNeTeC.2014.6995578.

[111] S. Raza, G. Huang, C.-N. Chuah, S. Seetharaman, J.P. Singh, MeasuRouting: a framework for routing assisted traffic monitoring, IEEE/ACM Trans. Netw. (TON) 20 (1) (2012) 45–56, doi:10.1109/TNET.2011.2159991.

[112] A. Wang, Y. Guo, F. Hao, T. Lakshman, S. Chen, Scotch: elastically scaling up SDN control-plane using vSwitch based overlay, in: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, CoNEXT'14, ACM, New York, NY, USA, 2014, pp. 403–414, doi:10.1145/2674005.2675002.

[113] D. Venmani, D. Zeghlache, Y. Gourhant, Demystifying link congestion in 4G-LTE Backhaul Using OpenFlow, in: 5th International Conference on New Technologies, Mobility and Security (NTMS), 2012, pp. 1–8, doi:10.1109/NTMS.2012.6208711.

[114] A. Passito, E. Mota, R. Bennesby, P. Fonseca, AgNOS: a framework for autonomous control of software-defined networks, in: IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014, pp. 405–412, doi:10.1109/AINA.2014.114.

[115] J. Li, S. Berg, M. Zhang, P. Reiher, T. Wei, Drawbridge: software-defined DDoS-resistant traffic engineering, in: Proceedings of the 2014 ACM Conference on SIGCOMM, 2014, pp. 591–592, doi:10.1145/2619239.2631469.New York, NY, USA

[116] S. Sun, L. Han, S. Cho, S. Han, J. Wang, B. Paillassa, Performance optimization of media distribution in overlay networks using OpenFlow, in: International Conference on Information Networking (ICOIN), 2014, pp. 276–281, doi:10.1109/ICOIN.2014.6799481.

[117] M. Shtern, R. Sandel, M. Litoiu, C. Bachalo, V. Theodorou, Towards mitigation of low and slow application DDoS attacks, in: IEEE International Conference on Cloud Engineering (IC2E), 2014, pp. 604–609, doi:10.1109/IC2E.2014.38.

[118] F. de Oliveira Silva, M. Goncalves, J. de Souza Pereira, R. Pasquini, P. Rosa, S. Kofuji, On the analysis of multicast traffic over the entity title architecture, in: 18th IEEE International Conference on Networks (ICON), 2012, pp. 30–35, doi:10.1109/ICON.2012.6506529.

[119] S. Laga, T. Van Cleemput, F. Van Raemdonck, F. Vanhoutte, N. Bouten, M. Claeys, F. De Turck, Optimizing scalable video delivery through OpenFlow layer-based routing, in: IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–4, doi:10.1109/NOMS.2014.6838378.

[120] X. Tu, X. Li, J. Zhou, S. Chen, Splicing MPLS and OpenFlow tunnels based on SDN paradigm, in: IEEE International Conference on Cloud Engineering (IC2E), 2014, pp. 489–493, doi:10.1109/IC2E.2014.20.

[121] H. Rodrigues, I. Monga, A. Sadasivarao, S. Syed, C. Guok, E. Pouyoul, C. Liou, T. Rosing, Traffic optimization in multi-layered WANs using SDN, in: IEEE 22nd Annual Symposium on High-Performance Interconnects (HOTI), 2014, pp. 71–78, doi:10.1109/HOTI.2014.23.

[122] R. Bennesby, E. Mota, P. Fonseca, A. Passito, Innovating on interdomain routing with an inter-SDN component, in: IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014, pp. 131–138, doi:10.1109/AINA.2014.21.

[123] R. Krishnan, D. Krishnaswamy, D. Mcdysan, Behavioral security threat detection strategies for data center switches and routers, in: IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2014, pp. 82–87, doi:10.1109/ICDCSW.2014.19.

[124] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, P. Castoldi, Effective flow protection in OpenFlow rings, in: Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013, pp. 1–3.

[125] A. Voellmy, H. Kim, N. Feamster, Procera: a language for high-level reactive network control, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN'12, ACM, New York, NY, USA, 2012, pp. 43–48.

[126] N. Foster, R. Harrison, M.J. Freedman, C. Monsanto, J. Rexford, A. Story, D. Walker, Frenetic: a network programming language, in: Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming 46 (9) (2011) 279–291, doi:10.1145/2034574.2034812.

[127] R. Alvizu, G. Maier, Can OpenFlow make transport networks smarter and dynamic? An overview on transport SDN, in: International Conference on Smart Communications in Network Technologies (SaCoNeT), 2014, pp. 1–6, doi:10.1109/SaCoNeT.2014.6867771.

[128] P. Pupatwibul, A. Banjar, A. Sabbagh, R. Braun, Developing an application based on OpenFlow to enhance mobile IP networks, in: IEEE 38th Conference on Local Computer Networks Workshops (LCN Workshops), 2013, pp. 936–940.

[129] S. Namal, I. Ahmad, A. Gurtov, M. Ylianttila, Enabling secure mobility with OpenFlow, in: IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–5.

[130] Y. Li, H. Wang, M. Liu, B. Zhang, H. Mao, Software defined networking for distributed mobility management, in: IEEE Globecom Workshops (GC Wkshps), 2013, pp. 885–889, doi:10.1109/GLOCOMW.2013.6825101.

[131] J. Schulz-Zander, N. Sarrar, S. Schmid, Towards a scalable and near-sighted control plane architecture for WiFi SDNs, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, in: HotSDN'14, ACM, New York, NY, USA, 2014, pp. 217–218, doi:10.1145/2620728.2620772.

[132] S. Yeganeh, A. Tootoonchian, Y. Ganjali, On scalability of software-defined networking, IEEE Commun. Mag. 51 (2) (2013) 136–141, doi:10.1109/MCOM.2013.6461198.

[133] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. McKeown, ElasticTree: saving energy in data center networks, in: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10, USENIX Association, Berkeley, CA, USA, 2010, p. 17.

[134] A. AL Sabbagh, P. Pupatwibul, A. Banjar, R. Braun, Optimization of the OpenFlow controller in wireless environments for enhancing mobility, in: IEEE 38th Conference on Local Computer Networks Workshops (LCN Workshops), 2013, pp. 930–935, doi:10.1109/LCNW.2013.6758534.

[135] J. Wang, X. Chen, C. Phillips, Y. Yan, Energy efficiency with QoS control in dynamic optical networks with {SDN} enabled integrated control plane, Comput. Netw. 78 (2014) 57–67.

[136] T. Pfeiffenberger, J.L. Du, Evaluation of software-defined networking for power systems, in: IEEE International Conference on Intelligent Energy and Power Systems (IEPS), 2014, pp. 181–185, doi:10.1109/IEPS.2014.6874175.

[137] N.M. Sahri, K. Okamura, Fast failover mechanism for software defined networking: OpenFlow based, in: Proceedings of The Ninth International Conference on Future Internet Technologies, CFI'14, ACM, New York, NY, USA, 2014, pp. 16:1–16:2, doi:10.1145/2619287.2619303.

[138] K. Phemius, M. Bouet, OpenFlow: Why latency does matter, in: IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 680–683.

[139] Z. Zhu, H. Li, K. Pan, C. Yu, F. Chen, D. Li, Centralized Flat Routing, in: International Conference on Computing, Management and Telecommunications (ComManTel), 2014, pp. 52–57.
[140] Z. Cai, A.L. Cox, T.S.E. Ng, Maestro: a system for scalable OpenFlow control, Technical Report, TSEN Maestro-Technical Report TR10-08, 2011.
[141] K. Nguyen, Q.T. Minh, S. Yamada, A software-defined networking approach for disaster-resilient WANs, in: 22nd International Conference on Computer Communications and Networks (ICCCN), 2013, pp. 1–5, doi:10.1109/ICCCN.2013.6614094.
[142] G. Sun, G. Liu, H. Zhang, W. Tan, Architecture on mobility management in OpenFlow-based radio access networks, in: IEEE Global High Tech Congress on Electronics (GHTCE), 2013, pp. 88–92, doi:10.1109/GHTCE.2013.6767247.
[143] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, A.W. Moore, OFLOPS: an open framework for OpenFlow switch evaluation, in: Proceedings of the 13th International Conference on Passive and Active Measurement, PAM'12, Springer-Verlag, Berlin/Heidelberg, 2012, pp. 85–95.
[144] B. Stephens, A.L. Cox, S. Rixner, Plinko: Building Provably Resilient Forwarding Tables, in: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, in: HotNets-XII, ACM, New York, NY, USA, 2013, pp. 26:1–26:7, doi:10.1145/2535771.2535774.
[145] S. Jeon, C. Guimarães, R.L. Aguiar, SDN-based mobile networking for cellular operators, in: Proceedings of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture, MobiArch '14, ACM, New York, NY, USA, 2014, pp. 13–18.
[146] T. Mahmoodi, S. Seetharaman, Traffic jam: handling the increasing volume of mobile data traffic, IEEE Vehicular Technol. Mag. 9 (3) (2014) 56–62.
[147] M. Borokhovich, L. Schiff, S. Schmid, Provable data plane connectivity with local fast failover: introducing OpenFlow graph algorithms, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, ACM, New York, NY, USA, 2014, pp. 121–126, doi:10.1145/2620728.2620746.
[148] S. Zhang, C. Kai, L. Song, SDN based uniform network architecture for future wireless networks, in: International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2014, pp. 1–5, doi:10.1109/ICCCNT.2014.6963056.
[149] A.Y. Ding, J. Crowcroft, S. Tarkoma, H. Flinck, Software defined networking for security enhancement in wireless mobile networks, Comput. Netw. 66 (2014) 94–101 (Leonard Kleinrock Tribute Issue: A Collection of Papers by his Students), doi:10.1016/j.comnet.2014.03.009.
[150] A. Lara, B. Ramamurthy, K. Nagaraja, A. Krishnamoorthy, D. Raychaudhuri, Cut-through switching options in a MobilityFirst network with OpenFlow, in: IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2013, pp. 1–6, doi:10.1109/ANTS.2013.6802877.
[151] N.A. Jagadeesan, B. Krishnamachari, Software-defined networking paradigms in wireless networks: a survey, ACM Comput. Surv. 47 (2) (2014) 27:1–27:11, doi:10.1145/2655690.
[152] M. Karimzadeh, A. Sperotto, A. Pras, Software defined networking to improve mobility management performance, in: Monitoring and Securing Virtualized Networks and Services, Lecture Notes in Computer Science, 8508,, Springer, Berlin/Heidelberg, 2014, pp. 118–122.
[153] F. Giust, M. Liebsch, Internet-draft—deployment of control-/data-plane separation in DMM, Technical Report, Internet Engineering Task Force (IETF), 2015.
[154] K. Sun, Y. Kim, Internet-draft—use case analysis for supporting flow mobility in DMM, Technical Report, Internet Engineering Task Force (IETF), 2015.
[155] H. Yang, K. Sun, Y. Kim, Internet-draft—routing optimization with SDN, Technical Report, Internet Engineering Task Force (IETF), 2015.
[156] S.-M. Kim, H.-Y. Choi, P.-W. Park, S.-G. Min, Y.-H. Han, OpenFlow-based Proxy mobile IPv6 over software defined network (SDN), in: IEEE 11th Consumer Communications and Networking Conference (CCNC), 2014, pp. 119–125.
[157] M. Ramadas, S. Ostermann, B. Tjaden, Detecting anomalous network traffic with self-organizing maps, in: Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection (LNCS), Springer-Verlag, 2003, pp. 36–54.
[158] M. Menth, M. Duelli, R. Martin, J. Milbrandt, Resilience analysis of packet-switched communication networks, IEEE/ACM Trans. Netw. 17 (6) (2009) 1950–1963, doi:10.1109/TNET.2009.2020981.
[159] S. Jain, K. Fall, R. Patra, Routing in a Delay Tolerant Network, in: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM'04, ACM, New York, NY, USA, 2004, pp. 145–158, doi:10.1145/1015467.1015484.
[160] C. Aikens, Facility location models for distribution planning, Eur. J. Operat. Res. 22 (3) (1985) 263–279.
[161] M. Pease, R. Shostak, L. Lamport, Reaching agreement in the presence of faults, J. ACM (JACM) 27 (2) (1980) 228–234, doi:10.1145/322186.322188.
[162] Open Networking Foundation, OpenFlow switch specification—version 1.3.0, 2012, https://www.opennetworking.org/sdn-resources/onf-specifications/openflow (accessed: August 2014).
[163] A. Marnerides, A. Schaeffer-Filho, A. Mauthe, Traffic anomaly diagnosis in Internet backbone networks: a survey, Comput. Netw. 73 (2014) 224–243. http://dx.doi.org/10.1016/j.comnet.2014.08.007.
[164] B. Han, V. Gopalakrishnan, L. Ji, S. Lee, Network function virtualization: challenges and opportunities for innovations, IEEE Commun. Mag. 53 (2) (2015) 90–97, doi:10.1109/MCOM.2015.7045396.

Anderson Santos da Silva is a M.Sc. student at the Federal University of Rio Grande do Sul (UFRGS), in Brazil. He obtained his B.Sc. degree in Computer Science also at Federal University of Rio Grande do Sul (UFRGS) in 2013. His current research interests include software-defined networking and network resilience.

Paul Smith is a Senior Scientist in the Safety and Security Department of the AIT, Austrian Institute of Technology. He received his Ph.D. in Computing from Lancaster University, UK in 2003. His research interests are in the area of ensuring the security and resilience of networked systems, in particular, those that support critical infrastructures, such as smart grids.

Andreas Mauthe is Reader in Networked Systems at the School of Computing and Communications (SCC), Lancaster University, UK. His research focus is within the networking and systems domain; main aspects are network management, autonomic networking architectures, network resilience mechanisms, and anomaly and malware detection in Cloud environments. He has co-authored more than 80 peer-reviewed papers and is also author of a textbook on Professional Content Management Systems. He worked in industry for more than four years in various management and research positions.

Alberto Schaeffer-Filho is an Associate Professor in the Institute of Informatics at Federal University of Rio Grande do Sul (UFRGS). Prior to that he was a Research Associate in Lancaster University for three years. He obtained his Ph.D. in Computing from Imperial College London in 2009. His research interests include network management, security and resilience of next generation networks. See http://www.inf.ufrgs.br/~alberto for selected papers.