Future Generation Computer Systems 122 (2021) 149–171

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges

Lubna Fayez Eliyan *, Roberto Di Pietro

Hamad Bin Khalifa University, College of Science and Engineering, ICT Division, Doha, Qatar

ARTICLE INFO

Article history:
Received 19 January 2020
Received in revised form 31 January 2021
Accepted 13 March 2021
Available online 26 March 2021

MSC:
00-01
99-00

Keywords:
SDN
Security
Attacks
DoS
DDoS
Research challenges
Survey

ABSTRACT

Software Defined Networking (SDN) is a new networking paradigm where forwarding hardware is decoupled from control decisions. It promises to dramatically simplify network management and enable innovation and evolution. In SDN, network intelligence is logically centralized in software-based controllers (the control plane), while network devices (OpenFlow Switches) become simple packet-forwarding devices (the data plane) that can be programmed via an open interface (OpenFlow protocol). Such decoupling of the control plane from the data plane introduces various challenges that include security, reliability, load balancing, and traffic engineering. Dreadful security challenges in SDNs are denial of service (DoS) and distributed denial of service (DDoS) attacks. For instance, in SDNs, DoS/DDoS attacks could flood the control plane, the data plane, or the communication channel. Attacking the control plane could result in failure of the entire network, while attacking the data plane or the communication channel results in packet drop and network unavailability. In this paper we deliver several contributions that shed light on the field of DoS/DDoS attacks in SDNs, providing a complete background about the area, including attacks and analysis of the existing solutions. In particular, our contributions can be summarized as follow: we review and systematize the state-of-the-art solutions that address both DoS and DDoS attacks in SDNs through the lenses of intrinsic and extrinsic approaches. Moreover, the discussed countermeasures are organized accordingly to their focus, be it on detection, mitigation, prevention, or graceful degradation. Further, we survey the different approaches and tools adopted to implement the revised solutions. Finally, we also highlight possible future research directions to address DoS/DDoS attacks in SDNs.

1. Introduction

Modern computer networks are typically built from a set of networking devices such as routers, switches, and firewalls. Such devices support many complex protocols as well as lists of commands and configurations based on a specific embedded operating system (OS) or firmware to achieve high throughput and performance. Hence, network administrators are limited to a set of predefined commands and should follow specific protocols and configuration policies to respond to a wide range of network events and applications. They have to manually transform these high-level policies into low-level configuration commands while adapting to changing network conditions. Often, they also need to accomplish these very complex tasks with access to very limited tools. Other limitations include vendor dependencies, complexity, inconsistent policies, and the inability to scale. As a result, network management and performance tuning are quite challenging, and thus error-prone [1]. It would be easier, simpler, and more efficient to support more protocols and applications if it were possible to program network controls in ways that are more responsive and flexible [2,3].

The idea of "programmable networks" has been proposed to facilitate network evolution. In particular, Software Defined Networking (SDN) is a new networking paradigm in which forwarding hardware is decoupled from the control decisions. It promises to dramatically simplify network management and enable innovation and evolution. In SDN, network intelligence is logically centralized in software-based controllers (the control plane), and network devices (OpenFlow Switches) become simple packet-forwarding devices (the data plane) that can be programmed via an open interface (OpenFlow protocol) [4]. Such SDN works as follows: An OpenFlow switch has one or more flow tables. These tables are used to control packets (e.g., forwarding or dropping) according to packet-handling rules (flow rules) received from a centralized controller. Therefore, according to the controller policy that manages flow tables, the OpenFlow switch can act as a router, switch, or firewall, or exhibit similar functions that depend on the packet-handling rules. Thus, the programmability

* Corresponding author.
E-mail addresses: leliyan@hbku.edu.qa (L.F. Eliyan), rdpietro@hbku.edu.qa (R. Di Pietro).

https://doi.org/10.1016/j.future.2021.03.011

of a network is realized through the control plane. When the OpenFlow switch receives a new flow, it checks for matches to its flow table entries. If there is a match, the action is applied; otherwise, the switch buffers the packet and then sends the packet-in message to the controller. Such message contains the headers of the packet to decide on and install the proper flow rule for the new flow. If the switch receives several new packet flows within a very short period, its buffer fills up. Then, the switch forwards the entire buffered packets to the controller, which could affect the control plane and add a communication overhead channel [1]. On the controller's side, receiving a high number of new flow requests exceeding the controller's processing capability could result in overwhelming the controller and making it possibly unreachable [5].

The idea of decoupling the control plane from the data plane in SDN provides many advantages to the networking environment but also introduces various challenges because of the programmable aspects involved. Such challenges include security, reliability, load balancing, and traffic engineering, to cite a few.

SDN security is more challenging than security in traditional networking, and among such challenges are both the denial of service (DoS) and distributed denial of service (DDoS) attacks [6]. Generally, DoS and DDoS attacks have become major threats to computer networks. In such attacks, by exhausting resources, several services are disabled, and the network's performance is downgraded. These attacks are considered successful when the attacker intentionally consumes resources that prevent hosts from using the targeted service. Different approaches can be used to perform DoS/DDoS attacks, including network-based approaches, such as flooding through TCP SYN, ICMP, or UDP packets, and host-based approaches, where one or several hosts target specific applications to exploit their memory structure, their authentication protocol, or a particular algorithm [3,7,8].

In SDNs, DoS/DDoS attacks could flood the control plane, the data plane, or the control plane bandwidth, while attacking the control plane of an SDN could result in failure of the entire network [9]. Indeed, the controller is considered the brain of the network, where it manages a large number of switches/applications. A DoS/DDoS attack on an SDN data plan could fill up the limited flow table memory of the OpenFlow switch. Once the switch memory is full, the switch is unable to receive new flows or to install new flow rules received from the controller, and hence packets are dropped. In this latter case, the switch is unable to forward buffered packets until there is free memory space in the switch flow table. Another target of DoS/DDoS attacks in the data plane is related to the generation of a massive number of new flows that do not match flow table entries defined by the switch. The switch buffers these packets before forwarding them to the controller through packet-in messages. However, if the rate of incoming new flows is high, the switch's buffer fills up, and then it forwards the entire packet to the controller instead of packet-in messages that contain headers only. This would result in a higher communication bandwidth consumption, as well as delays for the installation of new flow rules.

Therefore, there is a compelling need to provide a solution to mitigate DoS/DDoS attacks on SDNs. Such a solution should protect the communication channel between the OpenFlow switches and the SDN controller, handle packet-in messages properly, and maintain the functionality of forwarding legitimate traffic. Moreover, the envisaged solution should be able to discriminate between malicious and benign traffic in an efficient and effective manner, while maintaining an overall high performance.

Contributions: In this paper, we provide a thorough survey of state-of-the art solutions addressing both DoS and DDoS attacks in SDNs. In particular, we review and categorize the cited solutions based on their core techniques. Moreover, we also provide a detailed view of the components used in their implementation. Finally, we discuss possible countermeasures and related building blocks to DoS/DDoS attacks in SDNs.

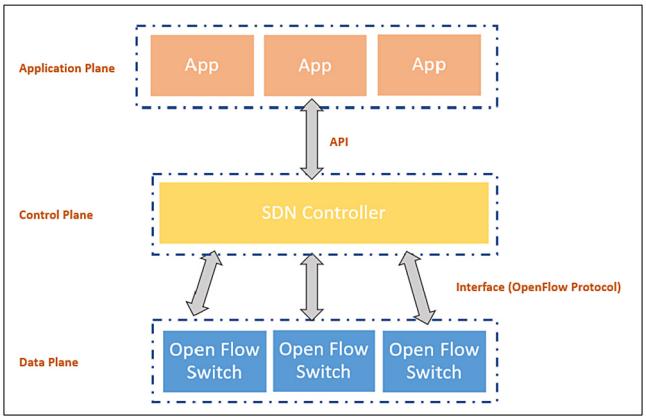


Fig. 1. SDN layers.

Roadmap: This paper is organized as follows: In Section 2 we discuss the background and related work about SDNs, including their architecture and operation modes, followed by DoS/DDoS attacks to SDNs. Examples of existing solutions to address DoS and DDoS attacks in SDNs, as well as the main components used in the corresponding implementation, are elaborated in Section 3. Then, in Section 4, we present an overview of the experimental results for the introduced approaches. The discussion of the provided solutions is reported in Section 5, while Section 6 draws some conclusions.

2. Background and related work

This section provides some background about different aspects of SDN, such as the SDN structure, operation mode, adopted open flow protocol, and how DoS/DDoS attacks target SDNs.

2.1. Software defined networks

The SDN architecture is composed of three main components: the application plane, the control plane, and the data plane, as shown in Fig. 1. The application plane contains SDN applications that are intended to perform various functionalities: enforcing security mechanisms and policies, performing network management, or running services on the SDN, to cite a few. Such applications communicate with the control plane via the northbound interface of the control plane [1,9]. The control plane contains the SDN controller, where the logic is centralized, as well as the global view of the network. Such a control plane manages the network, including applications in the application plane and the OpenFlow switches in the data plane. The third plane is the data plane, which is the combination of forwarding devices managed by the control plane through its southbound interface that implements the OpenFlow protocol [10]. The devices in the data plane are used to forward traffic flows based on specific flow rules.

The workflow implemented by the SDN OpenFlow switch follows: First, the OpenFlow switch compares the values of the incoming packet header with the headers field in the flow table. Once a match is found, a related action according to the flow table entry is applied. In case no match is found, a table-miss situation occurs. In this latter case, the OpenFlow switch forwards the information of the incoming packets to the controller through packet-in messages. These packet-in messages contain headers of new flows and are sent to the controller that, in turn, determines

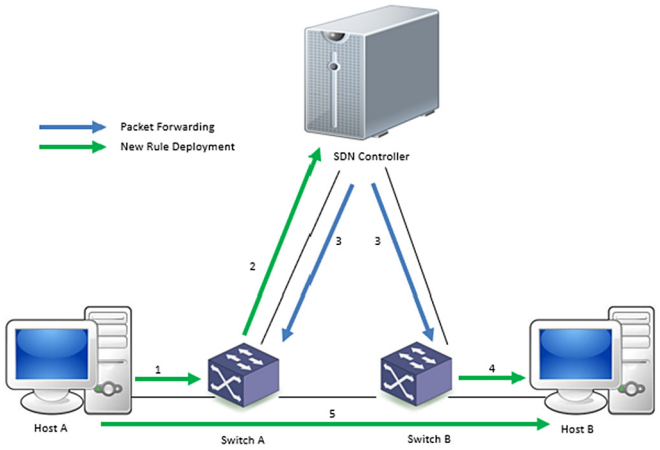


Fig. 2. SDN operation.

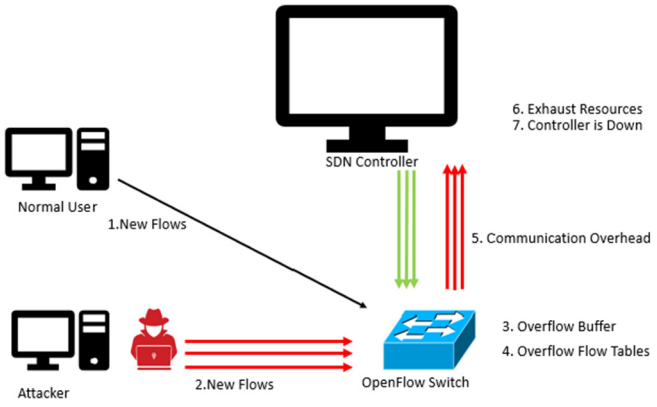


Fig. 3. DoS/DDoS attacks in SDNs.

the handling of the new packet by looking for a match in its own table. If no match is found, the packet is dropped; otherwise, the controller communicates with the OpenFlow switch through the OpenFlow protocol and installs a new entry flow table of the OpenFlow switch. Such new flow entry would instruct the OpenFlow switch about how to act on similar future packets without referring to the controller [11]. Fig. 2 shows the basic operation of the SDN.

2.2. DoS/DDoS attacks in SDNs

Some of the most common yet dreadful attacks on SDNs are DoS and DDoS attacks. Such attacks affect performance and network behavior. By exhausting resources, they disable or downgrade network services, and legitimate hosts are not able to communicate with the SDN controller or to send packets on the network [6].

DoS or DDoS attacks are achieved in SDNs by creating several new flows that flood the bandwidth of the control plane, the OpenFlow switches, and the SDN controller, which results in network failure for legitimate hosts. More specifically, attackers generate several new flows that have spoofed IP addresses but are sent from a single source (DoS) or multiple sources (DDoS). These spoofed addresses do not match any existing flow rules in the flow table of the OpenFlow switch, resulting in a table-miss situation. Such a situation results in generating massive packet-in messages sent to the SDN controller from the victim OpenFlow switch, which consumes communication bandwidth, memory, and CPU in both the control and the data plane of SDN [12]. Moreover, since the OpenFlow switch buffers packet-in messages before sending them to the controller, if several new flows are received within a very short time, the buffer fills up. This results in sending the entire packets of the new flow to the controller instead of sending the new flow's headers-only packet-in messages, thus resulting in higher consumption of the control plane's bandwidth and potentially delaying installation of new flow rules received from the SDN controller. Another situation in which massive new flows could result is filling up the forwarding table of the OpenFlow switch. As mentioned earlier, such a table includes different flow rules that direct the switch about forwarding the packets, and it is updated and managed by the controller [9]. Having several new flows results in installing new rules to the forwarding table of the switch. At some point, the forwarding table fills up, and therefore, upon receiving a new flow rule from the controller, it is unable to install it—hence dropping the packet and sending an error message to the controller [5]. Moreover, the switch would not be able to forward packets until there is free memory in its forwarding table, resulting in delays and dropping of incoming packets [13].

On the controller side, a high arrival rate of packet-in messages exceeding the controller processing capability results in overwhelming the controller and making it unreachable to legitimate traffic. This could result in the failure of the entire network, as the controller implements the logic of the SDN and manages the applications and the OpenFlow switches [13]. Fig. 3, shows a schematic view of a DoS/DDoS attack in SDNs.

2.3. Related work

To the best of our knowledge, literature surveys related to SDNs show that there is a need for further discussion of security issues and challenges for SDNs. For example, authors of [14] elaborate on challenges related to large-scale SDNs in terms of implementation, technologies deployed, and test-beds used. Nunes et al. [4] argue over different implementation alternatives of SDN architecture, as well as management of the OpenFlow protocol. Placement of multiple controllers in SDNs is studied in Zhang et al. [15] in terms of challenges, design principles, placement, performance, and architectures. Xia et al. [16] discuss the SDN layers in terms of implementation challenges, possible architectures, and performance enhancements.

Other works target fault management procedures in SDNs as [17] and [18]. The cited contributions propose various approaches for fault detection, prevention, localization, and correction along with the challenges in the adopted approaches. Lopes et al. [19] and Trois et al. [20] elaborate on programming paradigms of the SDNs in terms of performance and languages.

Rule placement is considered as a solution to manage resources in SDNs. In such solutions, some rules are assigned to OpenFlow switches to manage the limited resources while considering network configurations and policies as proposed in [21].

Zehra et al. [22] discuss various approaches of resource allocation problem in SDNs via the controller, and management of the resources based on requirements of the clients, while authors in [23] elaborate on load balancing approaches in SDNs to manage and respond to the incoming requests of the clients. Mathematical and verification tools of SDNs such as queueing theory and network calculus are discussed in [24], to help researchers in formalizing OpenFlow protocol operations; while authors in [25] elaborate on testing and debugging tools of the SDNs in terms of performance and capabilities. Other surveys review solutions that deploy SDNs to build IoT based systems for better management and isolation of the IoT devices as in [26] and [27].

An analysis of the literature that addresses security issues of the SDNs and the DoS/DDoS attacks targeting them, shows that there is a need to better highlight the details about the approaches implemented to address such attacks. What needs to be discussed in more detail are, for instance, the adopted methodologies, the used components, or the implementation equipment. Moreover, in the proposed works, authors map them into a single general approach such as analysis or use of entropy-based approaches, while the actual solution could involve multiple techniques at the same time. Therefore, in this paper, we study the different proposed solutions, and we elaborate in detail on the techniques used to realize them. This would provide a better understanding and enable building enhanced approaches based on the existing literature.

Examples of literature surveys related to the security of SDNs are in [8,9,28], and [29] where authors provide an overview of security challenges in SDNs for the different planes (i.e: application, control, and data). The authors also provide possible attacks and threats due to these challenges, as well as some approaches to address them. Yet, the authors do not have a special focus on a specific challenge or attack, while we do in this paper. Indeed, we focus specifically on DoS/DDoS attacks, related possible challenges, and countermeasures. Similarly, authors in [6] and [7] discuss challenges and solutions that address DoS/DDoS attacks in SDNs. For each solution, the authors elaborate on the advantages and disadvantages with an overview of the methodology adopted to implement the solution. Nevertheless, the authors do not discuss a variety of solutions as we do in this paper. Moreover, there are a few details about the solutions, and the adopted methodologies are missing. Researchers in [30] provide a detailed survey about approaches that employ SDNs to secure clouds against DoS/DDoS attacks. However, their survey does not tackle the problems in SDNs that cause the DoS/DDoS attacks: they rather show how to use SDNs to secure the clouds. In this survey, instead, we show and discuss the solutions of how to secure SDNs, rather than employing it in securing other applications. Similarly, authors of [31] provide a survey about how to use the SDNs to provide better security to the networks and their associated applications, rather than discussing the improvement of the SDN security against attacks and specifically against DoS and DDoS attacks, as we instead do in this paper. Similarly, authors of [32] show how to deploy SDNs in securing IoT based systems against various attacks such as replication, routing, or tempering attacks.

This paper aims to provide an overview of the recent approaches that address DoS/DDoS attacks in SDNs, while discussing details related to the adopted methodology, used components, and implementation elements. In addition, in this paper, we discuss the different works concerned with DoS/DDoS attacks and illustrate the contexts and a few scenarios in which these approaches can be implemented. Instead, we do not survey challenges related to SDNs layers (application, control, data), or their design aspects as already done in the previous surveys cited above. Similarly, we do not discuss all threats and attacks that SDNs are exposed to, rather, we focus in depth on one type of attack, which is the DoS/DDoS attack, probably accounting for the biggest threat to the ecosystem where SDNs are at use.

3. Solutions to address DoS/DDoS attack

DoS/DDoS attacks on SDNs are severe problems that need a real solution due to their effects on the performance of SDNs and connected hosts. According to [33], solutions that address DoS/DDoS attacks can be categorized as intrinsic solutions—the focus is on the components of SDNs and their functional elements – or extrinsic solutions – the focus is on network flows and their characteristics features. These two main categories can be also sub-categorized into further classifications as discussed in detail later in the next sections. The sub-categories of the intrinsic solutions are table-entry solutions, scheduling-based solutions, and architecture based solutions, while the extrinsic solutions are sub-categorized into statistics-based solutions and machine learning solutions. In this paper, the same categorization proposed in [33] is followed, enriched with additional categories such as making the OpenFlow switches intelligent or employing communication channels, as well as other non-categorized solutions.

Below we provide a discussion of the different classifications.

3.1. Intrinsic solutions

Intrinsic solutions are the approaches that focus on components of the SDNs and their functionality elements. These solutions are classified into table entry-based, scheduling-based, and architecture-based approaches. The subsections below elaborate on these three classes of intrinsic solutions and provide examples in adopting such approaches in addressing DoS/DDoS attacks in SDNs.

3.1.1. Table entry solutions

The table entry-based approaches propose solutions related to the limited size of tables for the OpenFlow switches. These tables could be buffering tables (memory) of unknown flows or forwarding tables that contain the flow rules. Each unknown flow needs a new entry in the switch memory—introducing a bottleneck during DoS/DDoS attacks, where packets with different IP addresses exist that generate massive table-miss flows. To address this problem, Mingxin Wang et al. [34] propose a mitigation approach that uses the cache of the OpenFlow switch to temporarily store all table-miss flows. As the cache size reaches a specific threshold, a decoupling process is applied to separate headers and payloads of packets and then classifies them based on certain attributes. Based on this classification, alert messages are sent to the SDN controller to apply further processing and to install needed rules to mitigate an attack. This approach would need to expand the SDN network in order to provide different cache equipment, as well as more processing cycles, to analyze the packet-in messages which might increase the experienced overhead. In addition, some concerns may arise in the event of a DoS/DDoS attack order of magnitudes greater than what the cache can handle, which may result in the inability to buffer processing packets; consequently, the entire network might fail. FloodDefender [12] is another solution to detect and mitigate DoS attacks on SDNs. It consists of different modules that have functionalities of management flows and table-miss packets. These modules are attack detection, table-miss management, filtration, and management of flow rules. Such functional modules operate together to identify traffic resulting from attacks and to perform the needed management actions to save the controller computational resources while removing useless switches' flow entries. An application-based solution to mitigate DoS attacks SDNs is proposed by Dridi et al. [35]. Such a solution is an application that reduces the overhead experienced by the table entries of the OpenFlow switches through direct communication with the controller. The proposed application's main task is to simultaneously manage different flows, aggregate rules, and collect different statistics for further analysis. Moreover, such an application reduces the overhead communication channel between the OpenFlow switches and the controller by setting different timeouts of installed rules for different traffic types based on collected statistics. These solutions impose greater overhead on the SDN network in terms of processing and management, yet,

they can be effective, in attack detection, mitigation, and flow management. The cited features can be achieved because the described solutions can manage several flows simultaneously, and collect further data to be able to detect the attack before it propagates into the network, hence further helping in preserving the network resources. Other solutions to mitigate DoS attacks are suggested by the researchers in [36]. Mainly, they suggest limiting the rate of incoming requests to the OpenFlow controller to keep it responsive and not over flooded. They also suggest reducing the timeout values of the flow table entries to prevent them from being flooded. These proposals may be categorized as lightweight enhancements to the SDN network, as they do not require an architectural alteration on the SDN. Nevertheless, these recommendations could introduce some challenges in their application, as they require modifications in the workflow of the SDN network and the OpenFlow protocol. Finally, Sandeep Singh et al. [37] aims to prevent and mitigate the launch of a DoS attack in its very first stages before it affects the network. In a nutshell, their idea is to buffer extra requests coming to the OpenFlow switch by controlling the queue size of the new flows in the switches. Under a heavy load, the size of the queue is increased. If the situation deteriorates and the switches are unable to handle the incoming requests, the source IP that is generating the high flow is blocked. Such a solution might be effective in controlling incoming requests and in protecting the network at the initial stages of starting a possible attack. Nevertheless, there should be a way to differentiate between the hosts that legitimately send packets at sustained rates and the attacking hosts, so as to maintain availability for the legitimate clients. Furthermore, special attention should be paid when it comes to the queue capacities, and the overhead in maintaining the queues in terms of optimal capacity and overhead in managing such queues.

Table 1 shows the main features addressed in the above mentioned works.

- Analysis Based Solution: this feature indicates that the proposed solution uses analysis of the incoming traffic to detect anomalies. This is achieved through monitoring the traffic, and collecting specific statistics related to its properties (e.g., number of packets per flow, number of requests within a time window). Based on the collected data, the statistics are then compared to a set of predefined thresholds to detect anomalies.
- Tools: such a feature indicates that the proposed solution uses some tools as Intrusion Detection Systems (IDS) or analysis tools for attack detection.
- Table-miss Management (Table_miss MGT): one of the approaches to lunch DoS/DDoS attacks is to overflow the memory of the OpenFlow switch with packets, triggering miss-match processes. Therefore, some solutions are based on managing table-miss situations of the new incoming flows, such that the SDN controller is protected before the attack propagates into the network.
- Flow Table Management (Flow Table MGT): this feature indicates that the proposed approach prevents the overflowing of the flow tables located in the OpenFlow switches. Such overflows occur when the controller installs new flow rules in the OpenFlow switches to route the huge number of the new incoming packets due to the attack. Approaches using this technique, therefore, rely on the management of the flow rules by eliminating or prioritizing the rules in such a way that the OpenFlow switch tables are protected from overflowing.
- Bandwidth Management (BW /Rate Limit): such a feature is rooted in bandwidth management. The purpose of this control is to limit the rate of incoming requests to the controller or the OpenFlow switch to protect them against DoS/DDoS attacks.
- Machine Learning (ML): it means that the approach involves machine learning mechanisms in attack detection.
- Controller Algorithm (Algo Ctrl): indicates that the approach deploys an algorithm that is running inside the SDN controller. The algorithm has access to some data within the SDN controller and uses these data to make decisions relevant to the prevention, detection, or mitigation of attacks.
- Multiple Controllers (MULT Ctrls): indicates that the approach is based on using multiple controllers that manage between each other to address the attack.
- Source Block on Detection: indicates that the solution traces the attacking source and then blocks it once it is detected.
- Flow Rules up on Detection: indicates that the solution installs flow rules to control the attack once it is detected. These floe rules are installed in the OpenFlow switches.
- Approach Type: such a feature shows whether the proposed solution is for attack prevention, detection, mitigation, or to continue working under attack situation.
- Implementation: such a feature shows the implementation methodology of the proposed solution. In particular, if it is based on Mininet emulation, hardware implementation, or virtual/ other software implementation methods.

3.1.2. Scheduling-based solutions

The second class of the intrinsic solutions is the scheduling-based solutions, which are solutions intended to protect the SDN controller from being overwhelmed and to manage the SDN resources. For example, [38], Hsu et al. provide a solution that limits the effect of DDoS attacks using hash functions associated with queues in the controller. The proposed solution's main idea is to distribute the incoming packets to the controller into multiple defined queues using the hash functions.

In turn, the controller serves these different queues in a round-robin fashion to avoid and limit failures of the OpenFlow switches due to the high traffic that could be DDoS malicious traffic. Such an approach limits the effect of the DoS/DDoS attacks on the network. Yet, the legitimate hosts could encounter delays in responding to their requests due to the scheduling procedures. Moreover, an overhead is introduced on the network which should be studied thoroughly to have the most beneficial trade-offs between the network size and its protection approach due to using the scheduling techniques. A similar approach is applied by Lim et al. [39] with similar advantages and disadvantages. Such a solution is based on defining multiple processing queues for each OpenFlow switch in the controller. Then, the controller schedules the serving of requests from the OpenFlow switches while limiting the effect of DDoS attacks by inserting delays between incoming requests. Therefore, overheads that could lead to failure of the controller can be avoided. As mentioned above, this approach could introduce delays to legitimate hosts as all of them are treated equally in terms of how their requested are handled. Other works provided an enhanced version of scheduling techniques to provide better responses to the legitimate hosts. This is accomplished through assigning priorities and trust values to the hosts as in Flow Ranger [40]. Mainly, it is a buffer-prioritizing solution that handles flow requests inside the SDN controller. In such a solution, different incoming requests are buffered to multiple queues in the controller and then served based on the priority values assigned to them. These priority values are assigned based on the trust values that are assigned to different hosts. Highly trusted hosts are assigned higher priority values and hence obtain better service in terms of requests response. Meanwhile, attacking hosts are assigned lower trust values, and hence, their requests are served with lower priority, which limits the effect of the DDoS attack. Such trust values are increased or decreased by the controller by maintaining a list of frequent

Table 1
Table entry-based solutions.

Paper	Analysis/ Statistics		Tools/IDS	Table_miss MGT	Flow Table	BW	ML	Algo Ctrl	MULT Ctrls	Approach				Source Block on Detect.	Flow Rules upon Detect.	Implementation		
	MonitorTraffic				MGT	Rate Limit				Prevention	Detection	Mitigation	Under attack			Virtual /Other SW	Mininet	HW
[34]	✓			✓			✓	✓	✓		✓	✓			✓		✓	
[12]	✓			✓	✓						✓	✓			✓		✓	✓
[35]	✓		✓		✓	✓					✓	✓					✓	
[36]	✓				✓	✓											✓	
[37]	✓					✓		✓		✓	✓			✓			✓	

Table 2
Scheduling-based solutions.

Paper	Analysis/statistics	Tools/IDS	Flow table	BW	Scheduling	Algo Ctrl	Approach				Mitigation		Implementation		
	Monitor traffic		MGT	Rate limit			Prevention	Detection	Mitigation	Under attack	Source block on detect.	Flow rules upon detect.	Virtual /Other SW	Mininet	HW
[38]					✓	✓				✓					✓
[39]					✓	✓				✓				✓	
[40]	✓				✓	✓				✓			✓		
[41]	✓		✓		✓	✓			✓	✓					
[42]	✓						✓			✓			✓		
[43]	✓		✓			✓				✓				✓	
[44]	✓	✓								✓		✓	✓		

users. Shoeb et al. [41] propose a solution to protect the controller based on assigning trust and threshold values to each connected host based on specific criteria. Based on the assigned values of trust and threshold, the controller selectively installs rules for new packet-in messages from hosts and assigns specific priorities to serve them. Therefore, the controller can survive under peak load under a DDoS attack situation, while introducing overhead on the network and the controller.

Some solutions are based on resource sharing between hosts. For example, Bedi et al. [42] use Deterministic Fair Sharing (DFS) to assign flows to different slots of the communication channel as a shared resource among them. The size of the slots depends on the channel size, the number of active flows, and the length of requests queued at the controller. Through monitoring the behavior of hosts using the channel, malicious flows are identified based on a probability value that increases as hosts use the channel unfairly. Hence, flows that are marked as malicious can be dropped to provide better service to the legitimate flows. Again, additional overheads are imposed on the network due to the monitoring processes, but genuine hosts benefit from a more efficient resource management and request processing, while at the same time the network resources are preserved. Adaptive Bubble Burst (ABB) [43] is another solution that is based on managing available resources under DDoS attack situations. Mainly, it takes advantage of the logically centralized architecture of SDN to allocate additional available resources dynamically to attacked applications. However, the proposed approach does not aim to detect or stop or defend against DDoS attacks. Rather, it enhances the availability of the resources that are being attacked. Finally, Fayaz et al. [44] assign available network resources using virtual machines to avoid bottlenecks in control and data plans. Mainly, the proposed solution is built for SDNs that employ virtual machines in their infrastructure. Using the resource manager component of virtualization functionality, available network resources are assigned dynamically, which provides more stability against attacks. Such a solution extends the SDN by adding the additional virtual machines, with extra management and supervision overheads. Yet, consideration should be given to the number of anticipated hosts and the possible magnitude of the attack before deploying it. In addition, there could be some vulnerabilities in virtual machines themselves, which is another point that should be assessed before employing them.

Table 2 provides information about the main features used in the different works above discussed. The description of new features that are not used in Table 1 are shown below:

- Scheduling: such a feature indicates that the solution uses scheduling of requests, packets, flows of the new incoming traffic, or scheduling incoming requests directed from OpenFlow switches to the SDN controller.
- Resource Sharing: such a feature indicates that the proposed approach manages the resources shared between the hosts or the OpenFlow switches. Some examples of these resources are communication channel bandwidth or processing cycles of the SDN controller.

3.1.3. Architecture-based solutions

The last class of intrinsic solutions is architecture-based solutions. Many of the solutions proposed in such a subcategory are based on decoupling the monitoring and controlling properties of the controller and extending the SDN architecture. For example, FloodGuard [13] introduces two modules to the SDN architecture, a proactive flow rule analyzer, and a packet migrator, to work under a DoS attack situation. The proactive flow rule analyzer is activated after the detection of a DoS attack to generate and install flow rules proactively into the OpenFlow switches, while the migration module is responsible for forwarding table-miss packets to the data plane's cache. Therefore, the two functionalities of controlling and monitoring are decoupled for a better reaction to an attack. Such a solution has no mechanism for recognizing the DoS/DDoS attack. It is designed to provide the network with stability and resistance to operate once the attack is detected. In light of what above, and taking into account the extension of the network architecture through the added modules, the solution could introduce some delays to the network response. Zhiyuan Hu et al. [45] propose a solution based on extending the architecture of the SDNs. Such a solution describes and adds functionality to the policies installed in the OpenFlow switches which are then used to redirect traffic. The key added features are policy characteristics, the role of policy maker, the level of security privileges, and the situation under which such policy is enabled. These new features help the SDN controller to obtain network policies from the northbound interface, that could be for instance, an automated security management module. The controller, in turn, converts these policies into flow entries and incorporate them into the switch flow tables. Therefore, these features allow access control management and roles for better reactions to attacks targeting SDNs. Nevertheless, according to the authors, the introduction of such a solution into existing commercial networks, such as carrier networks and mobile networks, is difficult and needs further research in order to implement it. FleXam [46] is another extension to the SDN architecture that extends the OpenFlow protocol to enable packet-level access by the SDN controller, which decides to sample specific packets of incoming traffic to identify botnets or malicious traffic. Moreover, the controller can send packet-based sampling to Intrusion Detection Systems (IDS) or other applications for analysis or security middle boxes. Hence, such an extension can provide protection to SDN from different attacks, including DDoS attacks. Nevertheless, when integrating it with other applications in the SDN, the solution could face some difficulties as it proposes an extension to the OpenFlow protocol that is built based on specific standards and regulations. Another extension of the SDN architecture, and particularly the OpenFlow protocol, is presented in [47]. In such an extension, flow entry can track and record the path of a flaw inside the network. The solution is composed of monitoring and analyzing packet-in messages to detect the attack and then trace back the attack source to block it. The tracking of the attack source would make the mitigation process more efficient anticipating and preventing the exhaustion of the network resources. Nevertheless, in terms of keeping track of these details, some considerations should be expressed about the size of the network, and the length of the path flows. Another point to be considered is how the network would react when experiencing attacks coming from a variety of sources, and how that would influence the network response time to legitimate hosts, while mitigating the attack source at the same time.

Other architecture-based solutions involve the use of multiple controllers that can handle each other's failures and balance the load among themselves. Such approaches, discussed below, share some of the key advantages and disadvantages. The key benefits are to include network stability and flexibility during large loads of legitimate requests or in attacking situations. These solutions primarily rely on preventing controller failure, as it is the brain of the entire SDN network. This is achieved by keeping at least one operational controller in the network. Thus, reliability and better response time in the network can be accomplished by providing backup controllers as well as load balancing among the controllers. Nonetheless, having multiple controllers causes several overheads in the network. That overhead is due to the management between the controllers in handling the requests, and the load-balancing procedures, and in selecting which controller should operate in case of failure. In addition, by sharing

the statuses between the various controllers and the details each controller has with other controllers in normal situations and during an attack, another overhead and issues are added. The works discussed below give some examples of solutions that include multiple controllers in their approach. For example, Bouet et al. [48] propose a solution that employs multiple distributed SDN controllers. Such controllers communicate with each other and share network-wide information among them. In case of failures due to attacks, the available distributed controllers manage the network and ensure its availability to connected hosts. Another mechanism is proposed by Fonseca et al. [49] to provide resilience to a network in case of controller failures due to DDoS attacks. Mainly, the mechanism defines a backup controller that acts once the primary controller fails. In order to have a seamless transition between two the controllers (from the host's point of view), the authors define a component that runs on top of the SDN architecture that can retrieve the last state of the failed controller and transfer its states and control to the new controller. Similarly, authors of works in [50] and [51] limit the effect of DDoS through the placement of multiple controllers. In [50], controllers work together and send signals to each other when under attack. Some of them can work as backup controllers for those that are under attack. Meanwhile, in [51] a cluster of controllers exchange messages and elect a leader. Such communication is needed to identify the controllers under attack and to eliminate the effects of network dependencies due to the attack while minimizing the effect of the DDoS attack on the network. Finally, Bahaa-Eldin et al. [2] use an additional controller along with the switch to form a sandbox-like model where malicious activities are performed in a controlled environment. In their proposed solution, all new flows' packets are forwarded to the sandbox to apply specific statistics and monitor them. Then, new rules are installed in the collaborative switch before being forwarded to the main SDN switch and the controller.

Other architecture-based solutions involve the use of Proof of Work techniques to verify the IP sources (hosts) that would like to connect to the SDN controller, as proposed in [52] and [53]. In such an approach, the attacker needs to spend a large number of resources to send the result to the controller in order to connect. The controller, in turn, can verify the result with low overhead and discard connection requests with the wrong results. Depending on the difficulty of the Proof of Work techniques, they could be very efficient in mitigating attackers as they require the attacker to spend resources to provide a correct solution to the puzzle sent by the controller. Nevertheless, legitimate hosts could be impacted as well in terms of spending some resources to connect to the network though to a smaller scale. Avant-Guard [54] uses another approach to verify IP sources. Mainly, it is based on verifying the handshake procedure of new flows. Those connections that complete the handshake procedure are authorized to connect to the controller. Additionally, inside the data plane of Avant-Guard, specific statistics are collected related to the rates of SYN requests, and these statistics are used to install new flaw rules based on the handled situations. Such a solution can prevent suspicious hosts from accessing the network, while simultaneously handling flows as directed by a relatively lightweight solution, based on collecting statistics and using them to support the decision process. Some solutions are based on hiding network hosts from attackers. For example, Jafarian et al. [55] propose the OpenFlow Random Host Mutation (OFRHM) technique to hide network hosts from external and internal attacks and scans. Such a technique is based on the controller allocating dynamics of random virtual IP addresses to the hosts. These IP addresses are obtained using a DCHP server connected to the controller, and then the translation to real IP addresses are done internally. Such an approach effectively hides the hosts' real IP addresses and protects them from direct attacks. Nevertheless, there is concern about having a situation in which multiple attackers are simultaneously requesting multiple hosts, in particular, how the DHCP server would be able to react and respond. Other solutions involve misleading the attacker about the private data network. This is achieved through the implementation of Moving Target Defense (MTD) algorithms to protect the network and to increase the attacker's cost, as in [56] and [57]. Such algorithms include the generation of random responses and payloads to mislead the attacker about the system's private data.

Finally, other solutions provide frameworks to be used in the detection or mitigation of DoS/DDoS attacks on SDNs. For example, Fresco [58] is a security framework for developing applications that can communicate with the OpenFlow switches. It provides a scripting language to write applications with enabled APIs. It is composed of a set of modules that work together simultaneously to enable the implementation of secure applications such as threat detection logic, event triggering, and data sharing. Khurshid et al. extend the Fresco [58] framework [59] by adding a layer between the controller and the OpenFlow switches. Such a layer monitors traffic dynamically to detect and identify any modifications or changes performed by the packet. Integrating such secure layers between the control and data planes enable more reliable protection to the control plan. Since the data or requests are pre-processed and examined until the controller manages them, and scanned for possible threats or suspicious activities. Nevertheless, such layers should be implemented carefully in terms of being sufficiently secure and not introducing vulnerabilities themselves, as well as to provide an efficient transfer of data between the control and data planes, while still providing responses in a short delay. Other researchers propose frameworks to be used by the developers to defend SDNs from DDoS attacks such as Lou et al. [60]. Such a framework enables developers to define detection and mitigation approaches on an abstract level through scripts that are then translated to flow rules in the switches.

Table 3 below shows the main features used in the proposed works above. An additional feature is added to the table that is related to the use of multiple controllers. The feature of using multiple controllers indicates that the solution is based on employing multiple controllers to respond to the DoS/DDoS attacks. Such multiple controllers act and operate with each other, could have a leader, or work as backup controllers in case of failures. Moreover, the multiple controllers could share the global state of the network with each other.

3.2. Extrinsic solutions

Extrinsic solutions focus on network flows and their features. They are classified as statistics or machine learning-based solutions [33]. The following subsections provide different examples of these solutions.

3.2.1. Statistics-based solutions

First, statistics-based solutions are other approaches that aim to protect the SDN controller by analyzing and collecting statistics related to flows. Kostas Giotis et al. [61], [41] apply statistical monitoring of flows. On the one hand, in [61], the solution employs the OpenFlow switches located on the edge of the network to collect different statistics. Then, these statistics are analyzed to detect abnormal traffic patterns. On the other hand, in [62], traffic is forwarded to the controller for statistics collection; the consequent analysis allows to detect and mitigate attacks. Similarly, Dong et al. [63] propose a solution based on statistics collection. In such a solution, statistics about incoming flows are collected and then used to classify flows as low-traffic or normal-traffic

Table 3
Architecture-based Solutions.

Paper	Analysis/ statistics	Clint Ver	Table_miss MGT	Flow table	BW	Scheduling	Algo Ctrl	MULT Ctrls	Approach				Mitigation		Implementation		
	Monitor traffic			MGT	Rate limit				Prevention	Detection	Mitigation	Under attack	Source block on detect.	Flow rules upon detect.	Virtual /Other SW	Mininet	HW
[13]			✓		✓	✓						✓		✓		✓	✓
[45]		✓		✓			✓										
[46]							✓										
[47]		✓					✓			✓	✓		✓				
[48]	✓						✓	✓				✓					✓
[49]								✓				✓				✓	
[50]								✓				✓					
[51]								✓				✓				✓	
[2]	✓							✓		✓	✓			✓		✓	
[52]		✓					✓		✓								
[53]		✓					✓		✓								✓
[54]	✓	✓							✓	✓	✓	✓		✓	✓	✓	✓
[55]							✓		✓							✓	
[56]							✓		✓								✓
[57]	✓								✓	✓	✓	✓		✓			✓
[58]																	
[59]									✓							✓	
[60]	✓						✓			✓			✓	✓			

flows. Low-traffic flows are flows that have few data packets yet lead to significant consumption of resources in the control plane, causing it to be unreachable. After the flow is classified, the attack detection process is applied through the use of the Sequential Probability Ratio Test (SPRT), which can detect interfaces compromised by a large number of low-traffic flows. Using statistics based methods to detect or mitigate the DoS/DDoS attacks is effective. It can be counted as a lightweight approach to know the status of the network and get aggregated information about the incoming requests and how the resources are consumed. However, some challenges might be faced in tuning the network based on specific threshold values that, if exceeded, could trigger attack mitigating processes.

Some solutions involve the development of monitoring/blocking applications that run on top of the SDN controller or third-party servers. For example, Thomas et al. [64] mitigate and detect DDoS attacks in SDNs using an application that is hosted on a third-party server. Such an application monitors and filters incoming traffic before forwarding it to the SDN controller. Based on collecting specific statistics related to flow rates with respect to specific conditions, a firewall can block incoming traffic from attacking hosts. Such a solution enables the protection of the control plane as well as mitigating the attack at the early stages, before exhausting the network resources. Yet, using a third-party server imposes some challenges related to the security of the server itself. Other application based solutions have been proposed by researchers. For example, Lim et al. [65] propose a DDoS-blocking application that runs on top of the SDN controller as well as a set of protected servers. The proposed application is designed to block botnets used to launch DDoS attacks. The detection mechanism is based on counting the incoming packets in protected servers and comparing such a value to a predefined threshold, while the defense mechanism is based on the redirection of protected server addresses once an attack is detected. Before clients are redirected to a new IP address on the server, they should solve a heavy computational barrier such as CAPTCHA [66]. For bots, it is difficult to solve problems such as CAPTCHA; hence, their flows and packets are eventually dropped. A later work presented by Lim et al. [65] is extended by the researchers in [67]. Mainly, the extended model does not involve the victim in detection and mitigation processes (protected servers); rather, all network-based functionalities are extended and used where all suspicious traffic is redirected to the CAPTCHA server. Trusted clients should be able to solve CAPTCHA to gain access to the network. The above presented works combine both the detection and the mitigation procedures to protect the SDN network. On the one hand, they keep track of the rate of the incoming packets to know the status of the resources utilization in the network, while on the other hand, they mitigate the attack through providing a complex puzzle to solve in order to retain access to the network resources; followed by a redirection to a new IP address. Yet, the mitigation procedure could result in a delay in handling the requests of the legitimate hosts. That is because of the dependencies required in the arrangement between the SDN controller and the protected servers as well as the need to solve the puzzle by the legitimate hosts. In addition, some parameter tuning should be performed carefully as per what concerns the threshold of incoming requests along with the number of times of solving the puzzle incorrectly to drop the connection by the controller.

The authors in [68] develop an algorithm that runs on the controller to detect an attack by analyzing traffic and comparing it against predefined thresholds. Upon attack detection, the controller updates the timeout values of the rules installed in the OpenFlow switches and directs the switches to drop specific packets. Similarly, works in [69,70], and [71] detect DDoS attacks based on exceeding predefined thresholds related to specific parameters. In the work presented in [69], the parameters are the number of packets sent by the source IP and of connections made by it. Meanwhile, in [70], the thresholds are related to network traffic and the frequency of occurrence of specific events. Once these thresholds are exceeded, certain flow rules are installed in the OpenFlow switches to mitigate the attack. Finally, the solution proposed in [71] measures the packet-in messages rate resulting from the table-miss process. This rate should be within a specific threshold value; if it is exceeded, a new rule is installed to redirect all table-miss packets towards the data plane's cache. Then, a filtration process is applied to the packets by the switches, and only the filtered packet-in messages are forwarded back to the controller. Indeed, similar to the approaches that are based on a predefined threshold for attack detection, careful tuning of both the threshold and the features should be performed. In particular, as per the cited features, they should be measured depending on the size of the network and the number of expected connecting hosts.

Other statistics-based solutions are based on capturing specific feature network flows to detect DoS/DDoS attacks. For example, [72] captures features related to the flow rate and flow volume of traffic. Such features are identified by installing specific flow rules in the OpenFlow switches in specific IP ranges. Such ranges dynamically change to shrink or expand depending on the detection of a DDoS attack. Upon detecting the source IP of an attack, specific rules are installed to drop traffic. Similarly, Dharma et al. [73] capture and analyze specific features to detect DDoS attacks. Their proposed solution is based on analyzing the destination address, traffic patterns, and the number of invalid packets per time window. Upon detection, new flow rules are installed in the OpenFlow switches for mitigation. Similarly, the contribution in [74] is based on analyzing traffic and then predicting spoofed IP addresses. This is achieved by means of mapping the MAC address, IP address, switch, and port. Once spoofed IPs are detected, an alarm is triggered, traffic coming from that source is traced, and the source is blocked. Another analysis-based approach is proposed by Chen and Chen [11]. In their proposed solution, a set of analyses is applied to flow table entries to detect attackers' reconnaissance attempts. Then, the attack is mitigated by providing false information about hosts in the network to attackers as well as activating a firewall to perform and apply automatic defense policies to protect the network. Another solution based on firewalling SDNs is in [75]. Thus, involving the firewall to block the suspicious hosts in the network offers a better mitigation process. Conversely, its integration with the SDN network poses some management-related challenges as well as challenges related to aligning the firewall with the SDN network in terms of communication standards and regulations.

Other solutions involve using tools to apply specific traffic analysis and collect statistics. Similar to the idea of integrating firewall within the SDN network, integrating other tools within the SDN network introduces some advantages and disadvantages. The advantages are associated with using components that are designed to perform particular and precise tasks related to analyzing the incoming requests or data. This offers more benefits, compared to programming an application running on top of the developer's SDN controller which could in itself introduce some vulnerabilities. Nevertheless, some problems could be met with the incorporation of these tools into the SDN environment in terms of compliance with the SDN protocols and standards. Further, additional overhead is placed on the network related to the management among the SDN controller and these tools. Examples of works that share the former advantages and disadvantages are discussed below. The proposed work in [76] uses the Bro tool [77] to analyze traffic to provide real-time detection of a DoS attack.

Based on the resulting analysis, new flow rules are installed in the OpenFlow switches for an immediate reaction to the attack. Other tools are used in other proposed approaches. For example, the authors in [78] use the sFlow tool [79], which is an open-source statistics- and time-based real-time traffic sampling tool for monitoring traffic at high speed. Based on applying specific analysis, a set of rules are installed in the OpenFlow switches to react to an attack. Similarly, Giotis et al. [62] use sFlow to detect DDoS attacks. Through sFlow's monitoring and collection of flow data, specific entropies are defined, such as destination IP address and destination port. Based on measuring the change in entropy values, specific rules are installed in the open flow switches to mitigate the attack.

Detection of DoS/DDoS attacks based on entropy variation is another statistical approach to protect the controller against attacks. There are some works that use entropy-based solutions. One of these is [80], where DDoS detection is applied in a distributed approach. In such an approach, entropies are defined and calculated in the OpenFlow switches located on the edge of the network. Therefore, the proposed solution reduces the controller overhead and enables a distributed approach to attack detection. Another solution based on entropy definition is proposed by Mousavi et al. [81]. In such a solution, entropy describes the destination IP address, and it is assumed that all hosts in the network have relatively equal traffic rates. Hence, based on the variations of entropy values among different hosts, the attack source can be identified. Using entropy-based approaches can be considered as a lightweight approach to gaining information about the status of the network, and the consumption of its resources. Yet, similar to the strategies involving threshold values, the entropy value should be carefully set and it should consider the network size and the number of hosts. This cited requirement is fundamental in providing precise decisions in the detection and mitigation procedures in the network.

Table 4 below shows the main features used in the different works discussed above, the description of the new added features compared to previous tables are the following:

- Client Verification (Client Ver): such a feature indicates that the solution verifies the client identity before enabling it to establish a connection with the SDN. Such verification could be achieved through several techniques such as Proof-of-Work, verifying TCP connections, and enforcing a complete handshake process.
- Firewall: such a feature indicates that the solution involves a firewall that is used for DoS/DDoS attack detection or mitigation.

3.2.2. Machine learning-based solutions

Machine learning-based solutions provide other approaches to identify both DoS and DDoS attacks on SDNs. In this kind of approach, the defense mechanism is a machine learning algorithm that is trained on attack free flows to be able to detect anomalies in traffic. For example, [82,83], and [84] use support vector machines (SVM) and neural networks (NN) to allow the controller to detect malicious DDoS traffic. Meanwhile, [85] uses the K-nearest neighbor (KNN) algorithm so the controller can detect and classify anomaly traffic. Some solutions involve hybrid approaches, such as [86], in which a (SVM) and a self-organizing map (SOM) are combined to detect and mitigate attacks. SOMs are also used in [87] and [66] to detect DDoS attacks. In [87], the SDN controller extracts flow features and feeds them into SOM; then, malicious flows are identified based on classification, and consequently dropped by the controller. Meanwhile, in [88], the solution is an application that runs on top of the SDN controller, where it can detect the true origin of an attack, as well as using SOM. In the above contributions, the detection is performed based on an adaptive and accurate classifier that performs decision making based on uncertain information. To realize that, supervised and unsupervised learning techniques are used to build a prediction model about the suspicious connections to the network. The support vector machines (SVMs) are used to learn a specific pattern within the data to help in producing an accurate classification. Though, these machine learning-based approaches take time to train and to produce the decision model in the network, and should be trained against a wide range of attacking patterns that could variate over the attacking period. Moreover, using such machine learning-based algorithms introduced some challenges in concurrently managing the SDN controller and the cited ML algorithms, hence likely imposing some delays in handling the legitimate requests.

Finally, some solutions involve the deployment of different algorithms and models to detect anomalous traffic patterns. For example, Mehdi et al. [89] show the implementation of the Threshold Random Walk with Credit-Based (TRW-CB), Rate-Limiting, Maximum Entropy Detect, and NETAD algorithms. Bing Wang [90] uses a graph model to detect malicious traffic patterns. Mainly, the model stores known traffic patterns as a relational graph. Once new traffic is generated, the model can determine whether it is malicious or not based on comparing graphs and hence block the source IP of malicious traffic. These algorithms, however, are based on flow grouping, clustering, and redirecting for classification and detection purposes, which are procedures that consume time and resources in processing and analysis to make the correct decision about the traffic anomalies.

Table 5 shows the main features used in the different works mentioned above related to the machine learning solutions.

3.3. Intelligent switches

Some solutions are based on making the OpenFlow switches more intelligent to be able to better detect or mitigate attacks. For example, Hyo-Bin Bae [91] proposes a solution to detect botnets that perform DDoS attacks on SDNs. Such a solution is based on detecting the spoofed IP addresses of botnets in the OpenFlow switches. The basic idea is that the attack packet has a mismatched source address; hence, the detection approach is based on matching the adjacent switch and ingress port information of the switch that sends the table-miss packet and then detects the spoofed IP source. Such a solution can be counted as a lightweight solution as it is based on simple data matching. Besides, applying the detection process at the switches helps in protecting the whole network and especially the controller at the early stages of the attack before it propagates in the network. However, depending on the attack size and the number of invalid requests, the switches could be flooded hence becoming unable to provide the detection process. This latter issue could lead to propagating the invalid requests to the controller, eventually resulting in a network failure.

Other solutions involve verifying the connection to the client to detect spoofed IP sources. For example, in [92], the OpenFlow switches verify hosts' connections to SDN and filter forged packets. Each real (unforged) connected host is assigned a specific score based on network resource usage. It should be mentioned that the behavior of the controller changes based on the scores assigned to the connected hosts. Therefore, the DoS attack effect is limited. Such a solution dynamically assigns network resources which is an advantage. Nonetheless, situations in which certain hosts within the SDN are attacked after verification of the link are not addressed in this proposed solution. Therefore, some hosts or services within the network are still vulnerable to the DoS/DDoS attacks.

Table 4
Statistics-based solutions.

Paper	Analysis/Statistics	Clint Ver	Tools/IDS	Flow table	FW	ML	Algo Ctrl	Traffic redirect to	Approach				Source block	Flow rules	Implementation		
	Monitor Traffic			MGT				Server/DB/SB/IDS	Prevention	Detection	Mitigation	Under Attack	on detect.	upon detect.	Virtual /Other SW	Mininet	HW
[61]	✓							✓		✓	✓			✓			✓
[62]	✓		✓				✓			✓	✓			✓		✓	✓
[63]	✓						✓			✓							
[64]	✓		✓		✓		✓	✓		✓	✓		✓	✓			
[65]	✓	✓						✓			✓		✓			✓	
[67]	✓	✓						✓			✓		✓			✓	
[68]	✓			✓			✓			✓	✓			✓		✓	
[69]	✓			✓			✓			✓	✓			✓	✓		
[70]	✓									✓	✓			✓	✓		
[71]	✓									✓	✓			✓			
[72]	✓									✓	✓			✓	✓		
[73]	✓					✓				✓	✓			✓			
[78]	✓					✓				✓	✓			✓	✓		
[74]	✓									✓	✓	✓				✓	
[11]	✓				✓					✓	✓						✓
[76]	✓		✓					✓		✓	✓			✓	✓		
[78]	✓		✓					✓		✓	✓			✓		✓	
[80]	✓									✓	✓		✓	✓		✓	✓
[81]	✓									✓						✓	

Table 5
Machine learning-based solutions.

Paper	Analysis/Statistics	BW	ML	Algo Ctrl	MULT Ctrls	Approach				Source Block	Flow rules	Implementation		
	monitor traffic	Rate limit				Prevention	Detection	Mitigation	Under attack	on detect.	upon detect.	Virtual /Other SW	Mininet	HW
[82]			✓	✓			✓						✓	
[83]	✓	✓	✓	✓	✓		✓	✓			✓		✓	
[84]			✓									✓		
[85]			✓	✓			✓	✓			✓	✓	✓	
[86]	✓		✓	✓			✓	✓		✓	✓	✓		✓
[87]	✓		✓	✓			✓					✓		
[88]			✓				✓			✓			✓	
[89]	✓	✓	✓	✓			#					✓		
[90]			✓				✓	✓		✓	✓		✓	

LineSwitch [93] is another solution that involves the switch intelligence to identify the DoS attack at the data layer. The idea is to make the OpenFlow switches that are located on the edges of the network to act as proxies of incoming TCP connections. It forces attackers to use their real IP addresses to be able to complete the three-way handshake process. Once the handshake process is completed, hosts can connect to the controller; otherwise, the switch blacklists them and blocks them for some time while monitoring their activity to prevent the attack from propagating into the network.

Finally, Zhang et al. [94] propose a solution to mitigate DoS attacks on SDNs using the port hopping technique. In such an approach, ports are randomly and dynamically mapped to unused ports. Hence, potential attackers are confused, and their attack cost is increased. Yet, the legitimate hosts spend more resources when using such a technique.

Table 6 provides information about the main features used in the above discussed solutions based on switch intelligence.

3.4. Secure communication channels

Other solutions involve securing the communication channel between data and control planes to address the DoS/DDoS attacks. For example, Ertaul et al. [95] employ Internet Protocol Security (IPsec), Advanced Encryption Standard (AES) encryption, and Transport Layer Security (TLS) to secure the communication channel. Using such well-defend and special purpose secure protocols provide more reliable protection of the network, rather than programming or developing new procedures that could introduce some vulnerabilities. However, integrating these protocols within the SDN environment is challenging because of the needed integration and arrangement between the cited protocols and the OpenFlow protocol—this latter one having its own standards and specifications. Other solutions employ the communication channel to reduce the overhead on the controller to protect it from failure due to the attacks. This can be achieved by limiting the rate of packet-in messages sent to the controller, as proposed in [5,96], and [97]. The use of the communication channel to address DoS/DDoS attacks helps to decrease the impact of the attack by reducing its intensity. Consequently, there are fewer incoming requests to the controller that could exhaust the network resources. Yet, depending on the attack size and its speed, the network would eventually fail in case of high attacks with a high rate of requests.

Table 7 shows the main features used when securing communication channels to address the DoS/DDoS attacks.

3.5. Other solutions

Some solutions include a combination of the different approaches above discussed, as proposed by Jeong et al. [98]. Mainly, application-level solutions limit DoS's effect through applying Deep Packet Inspection (DPI) process over incoming traffic to the controller, as well as scheduling and queuing of traffic. Traffic is redirected to the server before it is sent to the controller, where it is classified and assigned priority values to be scheduled. Moreover, the application allows communication between the data plane and the control plane to be limited, hence limiting the DoS effect. This solution combines several techniques such as inspecting the incoming packets, scheduling the requests and redirecting the flows. However, some packet delays occur as the controller has to wait for the DPI process before handling the requests to install the necessary flow rules leading to a delay in transmitting the traffic in the switches to the required addresses.

Some solutions involve protection against DoS attacks on SDNs that employ multiple virtual machines, such as in [99]. This is done by enforcing the isolation between shared resources and monitoring resource usage between shared hosts. Applying the concept of multiple virtual machines gives flexibility and adaptivity to the network by allocating and de-allocating resources as needed, particularly during attack times, where more resources can be provisioned. Furthermore, having isolated virtual machines helps in contaminating the attack if it occurred in one of the virtual machines. Yet, a huge amount of effort should be spent to maintain, manage, and orchestrate the different virtual machines

The authors in [100] detect a DDoS attack by leveraging throughput confidence interval. Such an interval is the normal distribution of throughput that is obtained in a normal situation where there is no DDoS attack. Based on the deviation from the confidence interval, an attack can be detected. Hence, it can be used to detect the attack as a lightweight approach. However, to make a precise decision, the confidence interval tuning should be carried out carefully considering the network size and the number of expected hosts joining. Finally, Li et al. propose DrawBridge [101], in which the end hosts can collaborate with the controllers to improve the detection of DDoS attacks. Mainly, they share a portion of traffic with the controller, which in turn deploys needed flow rules in switches. In such an approach, it is presumed that the hosts have successfully connected to the network, and that their sources are verified. Nonetheless, having a huge number of hosts sharing their flows with the controller will result in a server overload and potentially a server and network failure.

Table 8 discusses the main features used in the different works above mentioned.

4. Highlights on experimental works

In this section, we highlight some experimental results related to the previously discussed contributions. Some works use statistical analysis, reporting on the complexity and operating costs of handling attacks. Other works, instead, have specific contexts to run the experiment, with particular configurations and constraints. Finally, few other ones designed the testing environment based on the specific parameters that correspond to their implemented approach. For example, in machine learning approaches, experiments rely on the datasets and sampling rates used, as well as on the training sets. It is therefore very challenging to provide a standard testing environment and standard metrics to determine which solution is the most effective one. In light of the above, in this section, we focus on discussing the most compelling results of the proposed solutions, while pointing to specific resources for an in-depth analysis of the actual experimental setup and execution. Starting by the work in [34], researchers measured the amount of time needed by the controller to process and forward the first packet of the new flow during normal and attack situations. In normal situations, the proposed approach requires approximately 30 ms. However, in the case of an attack, while the traditional SDN network takes several orders of magnitude in terms of time to forward and process the packet, the proposed approach requires only about 310 ms. In [12], several measures have been applied to determine the efficiency of the proposed solution. The packet loss rate, for example, is calculated with an attack rate of 500 packet/seconds under attack situations. In the case of the OpenFlow protocol, the packet loss rate is around 95% when their proposed solution is not applied, while the packet loss rate does not exceed 5% when the solution is applied. Moreover, the researchers also assessed the efficacy of attack detection for the proposed solution. With less than a 5% false-positive rate, the proposed solution was able to identify precisely more than 96% of the attacking traffic. The researchers further measured the

Table 6
Switch intelligence-based solutions.

Paper	Analysis/Statistics monitoring traffic	Clint Ver	Algo Ctrl	Approach				Source block on Detect.	Implementation		
				Prevention	Detection	Mitigation	Under attack		Virtual IMP/ Other SW	Mininet	HW
[91]	✓				✓		✓	✓			
[92]	✓	✓	✓	✓						✓	
[93]		✓		✓						✓	
[94]				✓		✓				✓	

Table 7
Communication channel based solution.

Paper	Analysis/Statistics monitoring traffic	Flowtable MGT	BW Rate limit	Approach				Source Block on detect.	Flow Rules upon detect.	Implementation		
				Prevention	Detection	Mitigation	Under attack			Virtual Other SW	Mininet	HW
[95]				✓							✓	
[5]	✓	✓	✓	✓					✓		✓	
[96]	✓		✓	✓	✓	✓		✓			✓	
[97]	✓		✓	✓	✓	✓			✓	✓		

time delay under attacking situations. The average delay of the proposed approach at a rate of 500 packets/second is superior to the conventional OpenFlow solution, with values of 2038 ms and 18.7 ms, respectively. The packet loss rate was also measured in [38] to evaluate the proposed approach under attack situation for rates increasing from 2000 flows/second to 6000 flows/second. Without the proposed solution, when the peek attacking intensity approaches 2000 flows/second, the controller suffers from a client flow failure, indicating that the controller is overloaded. As the attacking rate increases to more than 2000 flows/second, the failure rate increases exponentially at the controller side, resulting in packets drop. With the implemented solution, however, reaching an attack rate of more than 5000 flows/second results in a failure for a fraction of less than 1% only. Similarly, the authors in [2] measured the packet drop rate to evaluate the proposed solution. This was accomplished by using a DDoS attack of 1000 packets/second targeting a host on the network. The results showed that the total number of packets dropped during the attack was almost zero, showing the efficacy of the proposed solution. Other researchers measured the accuracy and the ability to detect DDoS attacks, such as in [62] and [81]. In [62], the accuracy of the anomaly detection capability was measured for 100 Mbps of benign traffic injected with attack traffic at rates varying between 200 and 500 packets/seconds. A 100% detection accuracy was achieved while having 40% false positives rates in case of the DDoS attack. In [81], out of every 4000 packets sent to the controller, different percentages of them were attack packets. These percentages were 25%, 50%, and 75%, and were sent to the controller four times faster than normal traffic. In the case of a 25% rate, the detection success rate of the DDoS was 96%, while it approached 100% rate for 50% and 75% attacking rates. Additionally, the CPU consumption of the controller was measured with and without the proposed algorithm in case of a 25% attacking rate. The results showed no significant difference in the overhead generated by the proposed solution. Other researchers [43] examined the availability of a service hosted on an SDN network during an attack to evaluate the proposed solution. The scenario was set to send 500 requests to download an image hosted on a server. It was shown that only 4% of legitimate requests were completed without the proposed solution, compared to an 81% completion when the proposed solution was activated. In addition, the researchers measured the average packet processing time taken by the controller. The measurements were taken with around 1.4 million packets for each of the two cases—solution off and solution on. For the same system setup, including the same equipment, enabling the proposed solution resulted in an average processing time of 776 microseconds to process a packet, compared to 542 microseconds when the solution was disabled. This increase in packet processing time is due to statistics collection from switches, network delay, and response time from switches. The packet processing time by the controller was also measured in [53] in both normal and attack situations to evaluate the proposed solution that, in the normal situations, consumed an additional 50 microseconds. However, during the attack, the proposed solution took less than 200 microseconds to mitigate the attack, compared to around 500 micros seconds for a traditional controller. Similarly, researchers in [94] varied the packet size to measure the overhead of the proposed approach on CPU consumption. Varying the packet size between 0 and 50 KB resulted in an extra processing overhead comprised between 2.1% and 4.9%. Additionally, the authors measured the time and the number of packets used to perform a port scanning by the attacker using the Nmap scanning tool. The results showed, without having the proposed solution active, that it takes 6.7 s and 956 packets to detect the open ports at the tested server, compared to 29.4 s and 3103 packets when the proposed solution is off, showing the effectiveness of the proposed solution to protect the server. Researchers in [13] measured the impact of their proposed solution on the bandwidth between the connected clients in both benign and malicious situations. During normal situations, the proposed solution showed a negligible impact on the bandwidth. To assess how the solution copes with attacks, the attack magnitude was set between 130 and 150 packets/second, with a gradual increase. Without having the proposed solution enabled, at the rate of 130 packets/second the bandwidth of the network experienced a decrease of 50%. Later, increasing the attack rate to 500 packets/second, the whole network stopped working. However, with the proposed solution enabled, the bandwidth experienced just a negligible decay, even at the attack rate of 500 packets/second. Other researchers evaluated their solutions by calculating the introduced overhead for various packet rates, as in [46]. The overhead of the proposed solution was measured at 10, 100, and 1000 packets/second. The overhead was 0.1% for 10 and 100 packets/second, and 0.5% for 1000 packets/second. In addition, the authors also calculated the false-positive rate of the proposed solution, which was approximately 0.06% of the flows. Similarly, in [59], the overhead was measured to evaluate the proposed solution. This was achieved by measuring the effect of the proposed solution on the latency of the TCP connection as experienced by the SDN end-users. In the presence of the proposed solution, the latency of the TCP connection setup is increased by around 7% on average, compared to the case where the solution is not in operation. This rise is due to the additional modules used to strengthen the SDN network against attacks. Round Trip Time (RTT) is another metric that was used to evaluate some of the proposed solutions. For example, authors in [92], measured the round trip time (RTT) between the hosts during the attack with and without the proposed solution being turned on. With an attacking rate of 1000 packets/second, the RTT became extremely large compared to unchanged RTT in case of having the proposed solution enabled—RTT showed negligible decay in case of attack when the solution is enabled.

Table 8
Other solutions.

Paper	Analysis/ Statistics	Tools/IDS	BW	Scheduling	Algo Ctrl	Traffic Redirect to	Approach				Flow Rules upon Detect.	upon		
	Monitor Traffic		Rate Limit			Server/DB/SB/IDS	Prevention	Detection	Mitigation	Under attack		Virtual /Other SW	Mininet	HW
[98]	✓		✓	✓	✓	✓				✓			✓	
[99]							✓			✓		✓		
[100]	✓							✓					✓	
[101]					✓			✓						
[6]	✓	✓						✓	✓		✓			✓
[102]	✓	✓				✓		✓				✓		
[103]		✓				✓		✓	✓					✓
[104]		✓				✓		✓	✓			✓		

Table 9
Summary of highlighted experimental work.

Measurements/Tests		Attack situation – no proposed approach	Attack situation – with proposed approach
[34]	Average packet processing time by the controller	Several orders of magnitude of time	Approx. 310 ms
[12]	Packet loss rate at attacking rate of 500 packet/s	95% packet loss rate	<5%
	Attack detection efficiency		False-positive <5% Detection accuracy >96%
	Time delay at attacking rate of 500 packets/sec	2038 ms	18.7 ms
[38]	Packet loss rate at attacking rate increasing from 2000 flows/s to 6000 flows/s	Rate <2000 flows/sec: controller suffers from a client flow failure Rate>2000 flow/sec: complete packet drop	Rate >5000 flows/sec: <1% controller failure
[2]	Packet drop rate at attacking rate of 1000 packets/sec		Total packets dropped almost zero
[62]	Attack detection accuracy at attacking rates varying between 200 and 500 packets/s		False-positive <40% Detection accuracy: 100%
[80]	Attack detection accuracy at 25%, 50%, 75% attack traffic out of every 4000 packets sent to the controller		25% attack percentage: 96% detection accuracy 50%, 75% attack percentage: 100% detection accuracy
[43]	Service availability with an attack. set to send 500 requests	4% legitimate requests were completed	81% legitimate requests were completed
	Average packet processing time by the controller during attack	542 micro s/packet	776 micro s/packet
[53]	Average packet processing time by the controller during attack	500 microsec/packet	<200 micro s/packet
[93]	Time and number of packets used in port scanning till detecting the attack	29.4 secs and 3103 packets	6.7 secs and 956 packets
[13]	Impact on the bandwidth between the connected clients at attack rate of 130 and 150 packets/sec	130 packets/sec: <50 available 500 packets/sec: network stopped working	500 packets/secs: Negligible decay
[46]	Overhead at attacking rates of 10, 100, and 1000 packets/s		10 and 100 packets/secs: 0.1% 1000 packets/sec: 0.5%
	False-positive at attacking rates of 10, 100, and 1000 packets/s		Approx. 0.06%
[59]	Round trip time (RTT) among hosts at attacking rate of 1000 packets/s	Extremely large RTT	Unchanged RTT
	Available bandwidth at attacking rate of 1000 packets/s	Significant drop in bandwidth	Remains unchanged

Similarly, the available bandwidth at the same attacking rate remains unchanged in the case of the proposed solution enabled, compared to a significant drop when it is not enabled. Finally, in [50], the authors calculated the placement cost of multiple controllers on the network as well as the probability of DDoS attacks. There are various costs and probabilities for DDoS attacks, depending on both the number of controllers and the zones in which they are placed. The takeaway lesson of the cited contribution is that adding network controllers resulted in higher placement prices, but lowered the risks of DDoS attacks. Table 9 summaries the highlighted experimental work discussed above

5. Discussion, insights, and future research directions

In the following, we provide a categorization of the exposed solutions, discuss their pros and cons, and highlight some research directions.

Attack handling approach. Solutions that tackle the prevention of the DoS/DDoS attacks are relatively few compared to solutions that provide DoS/DDoS attacks detection or mitigation. Moreover, many of the detection and mitigation approaches would require extending the SDN architecture as the only way to address the DoS/DDoS attack problem in SDNs. Therefore, prevention of the attack, or providing the SDN with a graceful degradation procedure, is an area that needs more investigation in SDN security. Solutions that provide graceful degradation when under attack aim to maintain the availability of the network and provide fairness to the connected hosts in terms of shared buffers, handling requests, or bandwidth used. Moreover, such solutions that provide graceful degradation, aim to improve the reliability and scalability of the network through managing resources as in [38,39] and, [44]. Therefore, for the prevention and working under attack approaches, the main goal is to improve the QoS, while at the same time providing fairness to the connected hosts. Among the solutions that provide fairness to the connected clients and mitigate the DoS/DDoS attacks through monitoring behavior of the connected clients, we can cite [40] and [42].

Prevention-oriented approaches aim to prevent the attack before it could propagate to the network, such as in [52] and [53]. The cited solutions are based on either PoW techniques, or verification of the client's identity as in [59]. On the one hand, implementing such approaches (e.g. use of PoW) could be complex and costly. Additionally, these solutions could involve some overhead on legitimate clients. This overhead is because of the processing resources required to solve complex puzzles in PoW approaches. On the other hand, PoW based approaches reduce the required efforts in detecting and mitigating attacks while at the same time maintaining the availability of the network services.

Implementation. Depending on the complexity of the solution to address the DoS/DDoS attack, different components (e.g. controller, switch) of the SDN could be selected for accommodating the selected approach. For example, the solution could be implemented as an algorithm running inside the SDN controller, or as an application at the application layer of the SDN as in [34,37], and [46]. Using the SDN controller, or building an application on the application layer could generate an overhead for the network traffic, because of the needed flows redirection to the controller or to the application, as proposed in [48,64], and [98]. Additionally, solutions that involve running algorithms inside the SDN controller could result in a delay while responding to the legitimate clients by the controller because of the processing cycles consumed by the algorithm.

Other solutions could be implemented at OpenFlow switches. For example, some proposals use the OpenFlow switches located at the edges of the network. Therefore, the detection and mitigation of the attack could be achieved at the early stages, before propagating to the network, as in [91,92], and [93]. Yet, this could require additional components to be used by the switches as in [34], where an additional cache is used to process packets of new flows before sending them to the controller—while [13] uses the cache to avoid overflow of the switches' memory because of the attacks. In the latter proposed solution, the cache is used to temporarily store table miss packets before sending them to the controller with a limited rate, hence protecting the switch and the controller at the same time.

Other proposed solutions are implemented by adding an additional security layer on top of the control or data planes. Such a secure layer is integrated into the network and is responsible for maintaining the security of the controller and the network. That is achieved through dynamically monitoring the traffic to detect and identify any modifications or changes performed on the packets, as in [59]. Other approaches involve the addition of other components such as sandboxes, databases, or third-party services that are used for verification of flows as in [2,35], and [48]. However, the above-introduced solutions could involve modifications to the traditional SDN working flow and architecture because of the added hardware, while requiring more time for processing the requests before responding to the legitimate clients.

Techniques involved. Some of the proposed approaches involve relatively simple techniques such as defining entropy values and measuring specific features of network parameters (number of new flows, type of packet, number of packets sent by the source IP, or number of connections made by a specific source) as proposed in [69–71], and [85]. Mainly, measuring is the key in the statistical-based solutions as done in [61,62], and [63]. Other solutions could involve more complex techniques such as PoW [52,53], or machine learning techniques, such as in [82,83], and [84], where neural networks or SOM algorithms are used in the detection of the attack. Involving straightforward techniques simplifies the implementation of the solution and would require neither a major modification to the SDN architecture nor the addition of new components to be integrated into the network. However, these solutions could need some tuning, in terms of setting the values of the entropy values and might not be accurate compared to other techniques that involve verification of the clients' identity—such as in PoW based solutions. Machine-learning based approaches require training and testing data prior to implementing the solution, but it could provide accurate results for attacks' detection. Other solutions involve management of the SDN resources, as flow tables and buffers of the OpenFlow switches, processing cycles of the SDN controller, or allocated slots of the bandwidth in the network. Finally, some approaches employ the communication channel in limiting the effect of the attack on the network by controlling the consumed bandwidth by the switches and the clients through scheduling techniques.

Evaluation and testing. Most of the cited solutions were emulated and tested through the use of the Mininet simulator, such as in [13,35], and [49]. Some other solutions delivered a hardware implementation, such as in [48,53], and [62]. On the one hand, a Mininet simulator simplifies the implementation of the prototype and can be used as proof of the concept of the proposed solution. However, it imposes limitations on network configurations and sizes. On the other hand, a hardware implementation is another approach for evaluating and testing the solution. However, it is more challenging compared to Mininet in terms of both cost and configuring the components that should be compatible with each other in order to build the SDN and the testbeds for evaluating the solution.

The results of our research show that some areas need more investigations in order to address DoS/DDoS attacks in SDNs. Starting by the approaches for handling the attack, there are a few solutions that have a focus on preventing or working under attack circumstances, such as [13,37,42]. Other approaches focus on the detection or mitigation of the attack as for example in [12,34–36,38–44]. From what shown before, preventing the attack, and maintaining the network operational under attack is more important than detecting it or mitigating it. On the one hand, in the prevention process, the attack is prevented from propagating into the network before it could exhaust the network resources, and before triggering the need for the detection and the mitigation processes. On the other hand, the capability of continuing operating when under attack aims to provide a graceful degradation to maintain the availability of the network and to provide fairness to the connected hosts in terms of shared buffers, handling requests, or bandwidth used. Consequently, the availability, reliability, and scalability of the network are improved and the network can better survive under attack situations.

Another point that should be considered by future research is the use of a lightweight approach. These strategies are used mainly in statistically driven solutions where the network status can be identified because they can provide a measure of the resources consumed and the number of requests received—as in [41,61,64,68,70], and [74]. Depending on the complexity of the solution, machine learning-based approaches could be used providing accurate results for attack detection.

New research directions should also concentrate on approaches that verify the clients' identity as few works have a focus on such approaches in addressing the DoS/DDoS attacks as in [52] and [53]. Such approaches, like PoW based approaches, reduce the required efforts in detecting and mitigating attacks while at the same time maintaining the availability of the network services.

Another point that should be taken into consideration when addressing DoS/DDoS attacks in SDNs is related to the implementation. It seems safer to avoid extensions to the SDN architecture while developing SDN based solutions. Such extensions could include: adding extra hardware as caches and middleboxes, involving third-party servers, or adding middle secure layers as in [13,34], and [61]. There are a few reasons to avoid extensions. First, extending the SDN architecture would require revisiting the OpenFlow protocol which is the principal protocol in SDN. Such a protocol has two main bounds: north and south bounds which are mainly used to communicate the information between the application layer and the controller, and between the controller and the data layer respectively. Besides, such a protocol is having specific standards and rules that should be followed in the communication process. Hence, adding third-party servers, or other extensions would require to make the needed arrangement between the protocols they are using and the OpenFlow protocol that may need to be extended as well. Another reason to avoid extension is related to the overhead that results from the need for the redirection of the traffic to the third-party servers, or hardware. Such redirection processes would make the requests take more time, as they need to be buffered (at least in some circumstances) before they are analyzed and processed and then redirected again to the SDN controller. The last reason to avoid extensions is that these extensions might introduce security vulnerabilities. The ideal solution would be to develop an application that runs on top of the SDN controller and has direct communication with the controller to get the required information through the northbound of the OpenFlow protocol.

6. Conclusion

In this paper, we provided a comprehensive survey over the state-of-the-art solutions that address DoS and DDoS attacks in SDNs. Further, we discussed how countermeasures are adopted depending on whether the focus is on detection, mitigation, prevention, or graceful degradation. Additionally, we highlighted the implementation methods used to build the cited solutions, along with the techniques and the main features that should be considered when delivering a solution for the DoS/DDoS attacks on SDN. Moreover, we showed the possible evaluation methods to assess solutions for the cited attacks. Finally, we also highlighted possible research directions to tackle the DoS/DDoS attack problems in SDNs.

CRediT authorship contribution statement

Lubna Fayez Eliyan: Conceptualization, Methodology, Validation, Investigation, Writing - original draft. **Roberto Di Pietro:** Conceptualization, Methodology, Validation, Writing - review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

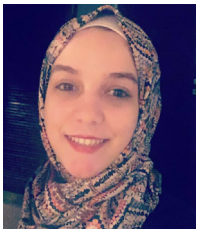
The authors would like to thank the anonymous reviewers for their suggestions, that helped to improve the quality of the manuscript. The publication of this article was funded by the Qatar National Library (QNL), Doha, Qatar and award NPRP 11S-0109-180242 from the Qatar National Research Fund (QNRF), a member of The Qatar Foundation. The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of QNL and QNRF.

References

[1] H. Farhady, H. Lee, A. Nakao, Software-defined networking: A survey, Comput. Netw. 81 (2015) 79–95.
[2] A.M. Bahaa-Eldin, E.E.E. Eldessouky, H. Dag, Protecting openflow switches against denial of service attacks, in: 12th International Conference on Computer Engineering and Systems, 2018, pp. 479–484.
[3] P. Bera, A. Saha, S. Setua, Denial of service attack in software defined network, in: 2016 5th International Conference on Computer Science and Network Technology, ICCSNT, 2016, pp. 497–501.
[4] B.A.A. 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. Tutor. 16 (3) (2014) 1617–1634.
[5] R. Kandoi, M. Antikainen, Denial-of-service attacks in OpenFlow SDN networks, in: Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, 2015, pp. 1322–1326.
[6] P.M. Ombase, N.P. Kulkarni, S.T. Bagade, A.V. Mhaisgawali, Survey on DoS attack challenges in software defined networking, Int. J. Comput. Appl. 173 (2) (2017) 19–25.
[7] G. Vaithianathan, R. Dheebalakshmi, Survey on solutions to DDoS attack to controllers in software defined networks, Int. J. Pure Appl. Math. 118 (2018) 1983–1990.
[8] S. Khan, A. Gani, A.W.A. Wahab, M.K. Khan, Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art, IEEE Commun. Surv. Tutor. 19 (1) (2017) 303–324.
[9] I. Ahmad, S. Namal, M. Ylianttila, A. Gurtov, Security in software defined networks: A survey, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2317–2346.
[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2008) 1–5.

[11] P.-J. Chen, Y.-w. Chen, Implementation of SDN based network intrusion detection and prevention system, in: 2015 International Carnahan Conference on Security Technology, ICCST, 2015, pp. 141–146.
[12] G. Shang, P. Zhe, X. Bin, H. Aiqun, R. Kui, FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9.
[13] H. Wang, L. Xu, G. Gu, FloodGuard: A DoS attack prevention extension in software-defined networks, in: 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, pp. 239–250.
[14] T. Huang, F.R. Yu, C. Zhang, J. Liu, J. Zhang, Y. Liu, A survey on large-scale software defined networking (SDN) testbeds: Approaches and challenges, IEEE Commun. Surv. Tutor. 19 (2) (2017) 891–917.
[15] Y. Zhang, L. Cui, W. Wang, Y. Zhang, A survey on software defined networking with multiple controllers, J. Netw. Comput. Appl. 103 (November 2017) (2018) 101–118.
[16] W. Xia, Y. Wen, S. Member, C.H. Foh, S. Member, D. Niyato, H. Xie, A survey on software-defined networking, IEEE Commun. Surv. Tutor. 17 (1) (2015) 27–51.
[17] Y. Yu, S. Member, X. Li, X. Leng, S. Member, L. Song, Fault management in software-defined networking : A survey, IEEE Commun. Surv. Tutor. 21 (1) (2019) 349–392.
[18] P. Fonseca, E.S. Mota, A survey on fault management in software-defined networks, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2284–2321.
[19] F.A. Lopes, M. Santos, R. Fidalgo, S. Fernandes, S. Member, A software engineering perspective on SDN programmability, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1255–1272.
[20] C. Trois, M.D.D. Fabro, L.C.E.D. Bona, M. Martinello, A survey on SDN programming languages : Toward a taxonomy, IEEE Commun. Surv. Tutor. 18 (4) (2016) 2687–2712.
[21] X.-n. Nguyen, D. Saucez, C. Barakat, T. Turletti, Rules placement problem in OpenFlow networks : A survey, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1273–1286.
[22] U. Zehra, M.A. Shah, A survey on resource allocation in software defined networks, SDN, in: 23rd International Conference on Automation & Computing, 2017, pp. 7–8.
[23] L. Li, Q. Xu, Load balancing researches in SDN: A survey, in; 7th IEEE International Conference on Electronics Information and Emergency Communication, ICEIEC, 2017, pp. 403–408.
[24] L. Girish, S. Rao, Mathematical tools and methods for analysis of SDN: A comprehensive survey, in: 2nd International Conference on Contemporary Computing and Informatics, IC3I, 2016, pp. 718–724.
[25] G.N. Nde, R. Khondoker, SDN testing and debugging tools : A survey, in: 2016 5th International Conference on Informatics, Electronics and Vision, ICIEV, IEEE, 2016, pp. 631–635.
[26] S. Bera, S. Misra, A. Vasilakos, Software-defined networking for Internet of Things : A survey, IEEE Internet Things J. 4 (2017) 1994–2008.
[27] N. Bizanis, F. Kuipers, SDN and virtualization solutions for the Internet of Things : A survey, IEEE Access 4 (2016) 5591–5606.
[28] S. Scott-hayward, S. Natarajan, S. Sezer, A survey of security in software defined networks, IEEE Commun. Surv. Tutor. 18 (1) (2016) 623–654.
[29] D.B. Rawat, S. Member, S.R. Reddy, Software defined networking architecture, security and energy efficiency : A survey, IEEE Commun. Surv. Tutor. 19 (1) (2017) 325–346.
[30] Q. Yan, F.R. Yu, Q. Gong, J. Li, Software-defined networking (SDN) and distributed denial of service (DDOS) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Commun. Surv. Tutor. 18 (1) (2016) 602–622.
[31] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking : A comprehensive survey, Proc. IEEE 103 (2014) 1–61.
[32] I. Farris, T. Taleb, S. Member, Y. Khettab, J. Song, S. Member, A survey on emerging SDN and NFV security mechanisms for IoT systems, IEEE Commun. Surv. Tutor. 21 (1) (2019) 812–837.
[33] K. Kalkan, G. Gür, F. Alagöz, Defense mechanisms against DDoS attacks in SDN environment, IEEE Commun. Mag. 55 (2017) 175–179.
[34] M. Wang, H. Zhou, J. Chen, B. Tong, An approach for protecting the openflow switch from the saturation attack, in: 4th National Conference on Electrical, Electronics and Computer Engineering, NCEECE 2015, 2016, pp. 729–734.
[35] L. Dridi, M.F. Zhani, SDN-Guard: DoS attacks mitigation in SDN networks, in: Proceedings - 2016 5th IEEE International Conference on Cloud Networking, CloudNet 2016, 2016, pp. 212–217.
[36] R. Kloti, V. Kotronis, P. Smith, OpenFlow: A security analysis, in: 21st IEEE International Conference on Network Protocols, ICNP, 2013, pp. 1–6.
[37] S. Singh, R.A. Khan, A. Agrawal, Prevention mechanism for infrastructure based Denial-of-Service attack over software Defined Network, in: International Conference on Computing, Communication and Automation, ICCCA 2015, 2015, pp. 348–353.
[38] S.W. Hsu, T.Y. Chen, Y.C. Chang, S.H. Chen, H.C. Chao, T.Y. Lin, W.K. Shih, Design a hash-based control mechanism in vswitch for software-defined networking environment, in: IEEE International Conference on Cluster Computing, ICCC, 2015, pp. 498–499.
[39] S. Yang, Y. Kim, H. Kim, S. Yang, S. Lim, Controller scheduling for continued SDN operation under ddos attacks, Electron. Lett. 51 (16) (2015) 1259–1261.
[40] L. Wei, C. Fung, FlowRanger: A request prioritizing algorithm for controller DoS attacks in Software Defined Networks, in: IEEE International Conference on Communications, vol. 2015, 2015, pp. 5254–5259.
[41] A. Shoeb, T. Chithralekha, Resource management of switches and Controller during saturation time to avoid DDoS in SDN, in: Proceedings of 2nd IEEE International Conference on Engineering and Technology, ICETECH 2016, 2016, pp. 152–157.
[42] H. Bedi, S. Roy, Mitigating congestion-based denial of service attacks with active queue management, in: Globecom 2013 - Communications QoS, Reliability and Modelling Symposium, 2013, pp. 1440–1445.
[43] D. Sattar, A. Matrawy, O. Adeojo, Adaptive Bubble Burst (ABB): Mitigating DDoS attacks in Software-Defined Networks, in: 17th International Telecommunications Network Strategy and Planning Symposium, 2016, pp. 50–55.
[44] S.K. Fayaz, Y. Tobioka, V. Sekar, M. Bailey, M. Bailey, Bohatei: Flexible and elastic DDoS defense, in: 24th USENIX Conference on Security Symposium, 2015, pp. 817–832.
[45] Z. Hu, M. Wang, X. Yan, Y. Yin, Z. Luo, A comprehensive security architecture for SDN, in: 18th International Conference on Intelligence in Next Generation Networks, ICIN 2015, 2015, pp. 30–37.
[46] S. Shirali-Shahreza, Y. Ganjali, Efficient implementation of security applications in OpenFlow controller with FleXam, in: IEEE 21st Annual Symposium on High-Performance Interconnects, HOTI 2013, 2013, pp. 49–54.
[47] R. Zhao, S. Wei, M. Ren, Combating DDoS attack with dynamic detection of anomalous hosts in software defined network, in: International Conference on Current Trends in Computer, Electrical, Electronics and Communication, CTCEEC 2017, 2018, pp. 37–42.
[48] M. Bouet, K. Phemius, J. Leguay, Distributed SDN for mission-critical networks, in: IEEE Military Communications Conference MILCOM, 2014, pp. 942–948.
[49] P. Fonseca, R. Bennesby, E. Mota, A. Passito, A replication component for resilient OpenFlow-based networking, in: 2012 IEEE Network Operations and Management Symposium, 2012, pp. 933–939.
[50] M.R. Haque, S. Ali, S.C. Tan, Z. Yusoff, L.C. Kwang, I.R. Kaspin, S.R. Ziri, Motivation of DDoS attack-aware in software defined networking controller placement, in: 2017 International Conference on Computer and Applications, ICCA 2017, 2017, pp. 36–42.
[51] R. Macedo, R. De Castro, A. Santos, Y. Ghamri-Doudane, M. Nogueira, Self-organized SDN controller cluster conformations against DDoS attacks effects, in: 2016 IEEE Global Communications Conference, GLOBECOM 2016, 2016, pp. 1–6.
[52] J. Li, T. Wolf, A one-way proof-of-work protocol to protect controllers in software-defined networks, in: 2016 Symposium on Architectures for Networking and Communications Systems - ANCS '16, 2016, pp. 123–124.
[53] T. Wolf, J. Li, Denial-of-service prevention for software-defined network controllers, in: 25th International Conference on Computer Communications and Networks, ICCCN 2016, 2016, pp. 1–10.
[54] S. Shin, V. Yegneswaran, P. Porras, G. Gu, AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks, in: ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, 2013, pp. 413–424.
[55] J.H. Jafarian, E. Al-Shaer, Q. Duan, Openflow random host mutation: Transparent moving target defense using software defined networking, in: 1st Workshop on Hot Topics in Software Defined Networks, HotSDN 2012, 2012, pp. 127–132.
[56] P. Kampanakis, H. Perros, T. Beyene, SDN-based solutions for Moving Target Defense network protection, in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2014, pp. 1–6.
[57] A. Aydeger, N. Saputro, K. Akkaya, M. Rahman, Mitigating crossfire attacks using SDN-based moving target defense, in: Conference on Local Computer Networks, LCN, 2016, pp. 627–630.
[58] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, FRESCO: Modular composable security services for software-defined networks, in: Network and Distributed System Security Symposium, NDSS 2013, 2013, no. February, pp. 1–16.
[59] A. Khurshid, W. Zhou, M. Caesar, P.B. Godfrey, X. Zou, W. Zhou, M. Caesar, P.B. Godfrey, Veriflow: verifying network-wide invariants in real time, 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI 13, 2013, pp. 15–27.
[60] S. Luo, J. Wu, J. Li, B. Pei, A defense mechanism for distributed denial of service attack in software-defined networks, in: 9th International Conference on Frontier of Computer Science and Technology, FCST 2015, 2015, pp. 325–329.

[61] K. Giotis, G. Androulidakis, V. Maglaris, A scalable anomaly detection and mitigation architecture for legacy networks via an OpenFlow middlebox, Int. J. Appl. Eng. Res. 9 (2015) 1958–1970.
[62] 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 (2014) 122–136.
[63] P. Dong, X. Du, H. Zhang, T. Xu, A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows, in: 2016 IEEE International Conference on Communications, ICC 2016, 2016, pp. 1–6.
[64] R.M. Thomas, D. James, DDOS detection and denial using third party application in SDN, in: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017, IEEE, 2018, pp. 3892–3897.
[65] S. Lim, J. Ha, H. Kim, Y. Kim, S. Yang, A SDN-oriented DDoS blocking scheme for botnet-based attacks, in: International Conference on Ubiquitous and Future Networks, ICUFN, 2014, pp. 63–68.
[66] L. von Ahn, M. Blum, N.J. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security, in: International Association for Cryptologic Research, 2003, pp. 294–311.
[67] T. Lukaseder, A. Hunt, C. Stehle, D. Wagner, R. Van Der Heijden, F. Kargl, An extensible host-agnostic framework for SDN-assisted DDoS-mitigation, in: Conference on Local Computer Networks, LCN, 2017, pp. 619–622.
[68] S. Wang, K.G. Chavez, S. Kandeepan, SECO: SDN secure controller algorithm for detecting and defending denial of service attacks, in: 5th International Conference on Information and Communication Technology, ICoIC7 2017, 2017, pp. 1–6.
[69] N.N. Dao, J. Park, M. Park, S. Cho, A feasible method to combat against DDoS attack in SDN network, in: International Conference on Information Networking, 2015, pp. 309–311.
[70] C. Yu Hunag, T. Min Chi, C. Yao Ting, C. Yu Chieh, C. Yan Ren, A novel design for future on-demand service and security, in: International Conference on Communication Technology, ICCT, 2010, pp. 385–388.
[71] H. Wang, L. Xu, A.G.G. Texas, OF-GUARD : A DoS attack prevention extension in software-defined networks motivation : Stopping the threats of DoS attack, 2014, pp. 2–3.
[72] Y. Xu, Y. Liu, DDoS attack detection under SDN context, in: IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.
[73] N.I. Dharma, M.F. Muthohar, J.D. Prayuda, K. Priagung, D. Choi, Time-based DDoS detection and mitigation for SDN controller, in: 17th Asia-Pacific Network Operations and Management Symposium: Managing a Very Connected World, APNOMS 2015, 2015, pp. 550–553.
[74] A. Hussein, I.H. Elhajj, A. Chehab, A. Kayssi, SDN security plane: An architecture for resilient security services, in: IEEE International Conference on Cloud Engineering Workshops, IC2EW 2016, 2016, pp. 54–59.
[75] M. Caprolu, S. Raponi, R. Di Pietro, FORTRESS: an efficient and distributed firewall for stateful data plane SDN, Secur. Commun. Networks 2019 (2019) 6874592:1–6874592:16.
[76] M.A. Lopez, O.C.M. Duarte, Providing elasticity to intrusion detection systems in virtualized Software Defined Networks, in: IEEE International Conference on Communications, 2015, pp. 7120–7125.
[77] S.B. Ambati, D. Vidyarthi, A brief study and comparison of, open source intrusion detection system tools, Int. J. Adv. Comput. Eng. Netw. (2013) 2106–2320.
[78] B.H. Lawal, A.T. Nuray, Real-time detection and mitigation of distributed denial of service (DDoS) attacks in software defined networking (SDN), in: 26th Signal Processing and Communications Applications Conference, SIU, 2018, pp. 1–4.
[79] P. Phaal, sFlow specification version 5, 2004, URL https://sflow.org/sflow_version_5.txt.
[80] R. Wang, Z. Jia, L. Ju, An entropy-based distributed DDoS detection mechanism in software-defined networking, in: 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015, 2015, pp. 310–317.
[81] S.M. Mousavi, M. St-Hilaire, Early detection of DDoS attacks against SDN controllers, in: 2015 International Conference on Computing, Networking and Communications, ICNC 2015, 2015, pp. 77–81.
[82] N. Meti, D.G. Narayan, V.P. Baligar, Detection of distributed denial of service attacks using machine learning algorithms in software defined networks, in: 2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI, 2017, pp. 1366–1371.
[83] S. Ezekiel, D.M. Divakaran, M. Gurusamy, Dynamic attack mitigation using SDN, in: 27th International Telecommunication Networks and Applications Conference, ITNAC 2017, 2017, pp. 1–6.
[84] R.T. Kokila, S. Thamarai Selvi, K. Govindarajan, DDoS detection and analysis in SDN-based environment using support vector machine classifier, in: 6th International Conference on Advanced Computing, ICoAC 2014, 2015, pp. 205–210.
[85] A. Prakash, R. Priyadarshini, An intelligent software defined network controller for preventing distributed denial of service attack, in: Second International Conference on Inventive Communication and Computational Technologies, ICICCT, 2018, pp. 585–589.
[86] T.V. Phan, N.K. Bao, M. Park, A novel hybrid flow-based handler with DDoS attacks in software-defined networking, in: 13th IEEE International Conference on Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress, 2017, pp. 350–357.
[87] R. Braga, E. Mota, A. Passito, Lightweight DDoS flooding attack detection using NOX/OpenFlow, in: 35th Annual IEEE Local Computer Networks, LCN, 2010, pp. 408–415.
[88] T. Wang, H. Chen, SGuard: A lightweight SDN safe-guard architecture for DoS attacks, China Commun. 14 (2017) 113–125.
[89] S. Mehdi, J. Khalid, S. Khayam, Revisiting traffic anomaly detection using software defined networking, Lecture Notes in Comput. Sci. 6961 (2011) 161–180.
[90] B. Wang, Y. Zheng, W. Lou, Y.T. Hou, DDoS attack protection in the era of cloud computing and Software-Defined Networking, in: 22nd International Conference on Network Protocols DDoS, 2015, pp. 308–319.
[91] H.-b. Bae, M.-w. Park, S.-h. Kim, T.-m. Chung, Zombie PC detection and treatment model on software-defined network, in: Computer Science and Its Applications, 2015, pp. 837–843.
[92] M. Zhang, J. Bi, J. Bai, Z. Dong, K. Gao, Z. Li, X. Yin, FloodShield : Securing the SDN infrastructure against denial-of-service attacks, in: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE, 2018, pp. 687–698.
[93] M. Ambrosin, M. Conti, F. De Gaspari, R. Poovendran, Lineswitch: Tackling control plane saturation attacks in software-defined networking, IEEE/ACM Trans. Netw. 25 (2017) 1206–1219.
[94] L. Zhang, Y. Guo, H. Yuwen, Y. Wang, A port hopping based DoS mitigation scheme in SDN network, in: 12th International Conference on Computational Intelligence and Security, CIS 2016, 2017, pp. 314–317.
[95] L. Ertaul, K. Venkatachalam, N. Star, Security of Software Defined Networks (SDN), in: ICWN'17 - the 16th Int'L Conf on Wireless Networks, 2017, pp. 24–30.
[96] M. Kuerban, Y. Tian, Q. Yang, Y. Jia, B. Huebert, D. Poss, FlowSec: DOS attack mitigation strategy on SDN controller, in: IEEE International Conference on Networking Architecture and Storage, NAS 2016, 2016, pp. 7–8.
[97] A.F.M. Piedrahita, S. Rueda, D.M. Mattos, O.C.M. Duarte, FlowFence: A denial of service defense system for software defined networking, in: 2015 Global Information Infrastructure and Networking Symposium, GIIS 2015, 2015, pp. 1–6.
[98] S. Jeong, D. Lee, J. Hyun, J. Li, J.W.K. Hong, Application-aware traffic engineering in software-defined network, in: 19th Asia-Pacific Network Operations and Management Symposium: Managing a World of Things, APNOMS 2017, 2017, pp. 315–318.
[99] D.M.F. Mattos, O.C.M.B. Duarte, XenFlow: Seamless migration primitive and quality of service for virtual networks, in: 2014 IEEE Global Communications Conference, GLOBECOM 2014, 2014, pp. 2326–2331.
[100] A. Sangodoyin, B. Modu, I. Awan, J. Pagna Disso, An approach to detecting distributed denial of service attacks in software defined networks, in: 6th International Conference on Future Internet of Things and Cloud, FiCloud, 2018, pp. 436–443.
[101] J. Li, S. Berg, M. Zhang, P. Reiher, T. Wei, Drawbridge: software-defined DDoS-resistant traffic engineering, in: ACM Conference on SIGCOMM, 2014, pp. 591–592.
[102] T. Ha, S. Yoon, A.C. Risdianto, J.W. Kim, H. Lim, Suspicious flow forwarding for multiple intrusion detection systems on software-defined networks, Netw. Forensics Surveillance Emerg. Netw. 30 (6) (2016) 22–27.
[103] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, Z. Zhang, Enabling security functions with SDN: A feasibility study, Comput. Netw. 85 (2015) (2015) 19–35.
[104] T. Chin, X. Mountrouidou, X. Li, K. Xiong, Selective packet inspection to detect DoS flooding using software defined networking (SDN), in: IEEE 35th International Conference on Distributed Computing Systems Workshops, ICDCSW 2015, 2015, pp. 95–99.



Lubna Fayez Eliyan is a Ph.D. student in Computer Science and Engineering program at HBKU-CSE in Doha–Qatar, with a major in cybersecurity. She holds a B.Sc. degree in computer engineering from Qatar University, and a M.Sc. in computer networks from the same university. Her main research interests include computer networks and security and privacy of networks, IoT, and crowd simulation for studying human behavior.

Dr. Roberto Di Pietro, ACM Distinguished Scientist, is Full Professor in Cybersecurity at HBKU-CSE. Previously, he was in the capacity of Global Head Security Research at Nokia Bell Labs, and Associate Professor (with tenure) of Computer Science at University of Padova, Italy. He has been working in the security field for 23+ years, leading both technology-oriented and research-focused teams in the private sector, government, and academia (MoD, United Nations HQ, EUROJUST, IAEA, WIPO). His main research interests include AI driven cybersecurity, security and privacy for wired and wireless distributed systems (e.g. Blockchain technology, Cloud, IoT, OSNs), virtualization security, applied cryptography, computer forensics, and data science. Other than being involved in M&A of start-up – and having founded one (exited) – he has been producing 230+ scientific papers and patents over the cited topics, has co-authored three books, edited one, and contributed to a few others. In 2011–2012 he was awarded a Chair of Excellence from University Carlos III, Madrid. In 2020 he received the Jean-Claude Laprie Award for having significantly influenced the theory and practice of Dependable Computing.