Journal of Network and Systems Management (2023) 31:43
https://doi.org/10.1007/s10922-023-09732-5

On Profiling, Benchmarking and Behavioral Analysis of SDN Architecture Under DDoS Attacks

Nguyen Huu Thanh[1] · Nguyen Ngoc Tuan[1] · Dang Anh Khoa[1] · Le Cong Tuan[1] · Nguyen Trung Kien[2] · Nguyen Xuan Dung[1] · Ngo Quynh Thu[1] · Florian Wamser[3]

Received: 26 July 2022 / Revised: 20 December 2022 / Accepted: 9 March 2023

Abstract
Software-Defined Networking (SDN) has attracted much attention from research and industrial communities recently as it is more agile and flexible compared to conventional networking technology in offering new network functions and services. By separating the network control functions from the forwarding devices and placing them in a centralized, softwarized and programmable SDN controller, new network functions and services can be easily added into the network in an on-demand manner. However, the centralized control paradigm and the flow-based forwarding principle make the SDN architecture more fragile and vulnerable to malicious actions, such as cyber hijacking or DDoS attacks. In this paper, we focus on analyzing and evaluating negative impacts of DDoS attacks on the SDN architecture. By performing stress tests, the performance of such common SDN controllers as POX, Ryu and Floodlight under DDoS attacks is benchmarked, along with their impacts on the SDN switch and OpenFlow channel. We also address some new threats and vulnerabilities introduced by the nature of SDN.

Keywords SDN · DDoS · Security · SDN Performance · OpenFlow

1 Introduction

Software-Defined Networking (SDN) is an emerging network architecture [1, 2] supposed to overcome the limitations of conventional network technologies. The control and management functions of SDN are separated from its forwarding functions

Nguyen Ngoc Tuan, Dang Anh Khoa, Le Cong Tuan, Nguyen Trung Kien, Nguyen Xuan Dung and Ngo Quynh Thu have contributed equally to this work.

✉ Nguyen Huu Thanh
thanh.nguyenhuu@hust.edu.vn

Extended author information available on the last page of the article

Published online: 29 March 2023

and placed in the centralized, softwarized and programmable *SDN controller*. This enables a number of new possibilities and advantages compared to the conventional distributed networking paradigm.

Firstly, since complex control functions reside in the centralized SDN controller, SDN-enabled commodity and generic forwarding devices can be deployed instead of vendor-dependent, expensive routers and switches, thus reducing the capital expenditure (CAPEX) invested in the network infrastructure. Secondly, the SDN architecture is more flexible and agile as it allows defining *network flows* flexibly based on a set of tuples extracted from MAC, IP and TCP/UDP headers, thus specific forwarding rules and policies for individual flows can easily be performed. Moreover, network administrators can use programming languages to add on-demand functionalities in the network, such as traffic monitoring, routing, QoS guarantees and security functions and so forth.

However, the aforementioned advantages do not come for free. In the traditional distributed networking paradigm with coarse-grained destination-based packet forwarding, every router processes packets independently by finding next hops to the destinations. The number of states that a router should maintain in its forwarding table is mere the number of destination networks. Contrarily, in SDN flow-based forwarding paradigm, if the tuples of an arriving packet do not match, the packet is treated as a new flow, which requires the controller to create and install new rules in the forwarding table of the SDN-enabled switch. The finer-grained the flows are defined, the more rules the controller should handle and install in the switch. In case the number of new incoming flows is large, the controller can be overwhelmed by processing new flows.

Thus, despite the improvement that SDN can provide, some vulnerability issues are of great concerns as the follows: (1) the SDN controller in the centralized architecture could be the single-point-of-failure that can have an impact on the overall network operations; and (2) the reactive mechanism of SDN to install new rules could overwhelm both the controller and SDN switches due to their limited processing, bandwidth, memory or flow table capabilities. These vulnerabilities tolerate a number of approaches attempting to attack functional blocks in the SDN architecture as well as hosts residing within the SDN domain, such as *cyber hijacking* and *Distributed Denial of Service* (DDoS).

In DDoS attack [3, 4], attackers take control of a large number of hosts in the Internet, known as *botnets* and use them to send extremely large number of packets to the intended victims. Consequently, the destination victim with limited resources is overloaded and cannot provide service to the legitimate users. As shown in Fig. 1, DDoS can be classified into three different types of attacks, namely:

- In *volume-based attacks*, the attacker sends a huge amount of traffic to saturate the bandwidth of the victim. UDP flood, DNS amplification and ICMP flood are good examples of this attack type. In traditional networks, the high-volume traffic generally does not affect network devices, i.e., routers and switches but rather the data paths interconnecting them. However in SDN, the saturation of the controller–switch links caused by a volume-based attack could disrupt the communication between the centralized controller and its forward-

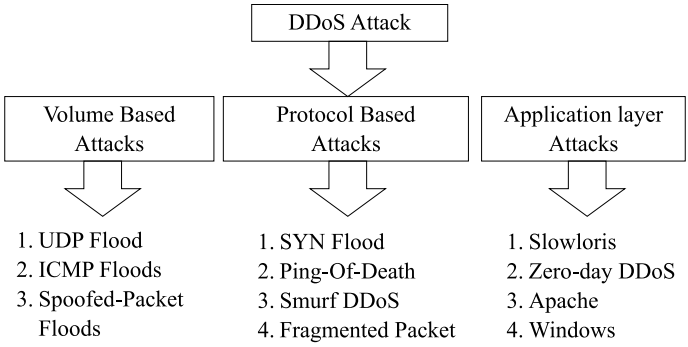


Fig. 1 DDoS attacks classification [3]

ing devices and lead to the malfunctioning of the whole network. Moreover, depending on the number of flows carrying the traffic volume, the controller may be overloaded by processing or buffering new flow requests.

- *Protocol-based attacks*, also known as a *state-exhaustion attacks*, cause a service disruption by consuming all the available state table capacity of servers or intermediate resources like firewalls and load balancers. Protocol-based attacks utilize weaknesses in Layer 3 and Layer 4 of the OSI model to make the target inaccessible. Besides the negative impact on end devices, this type of attack, such as TCP-SYN flood, could be one of the most malicious attacks against the SDN architecture. The large number of new incoming flows can trigger the controller to handle a large number of requests, causing it to overload.
- *Application layer attacks* refer to DDoS attack that target Layer 7 in the OSI model, in which it is designed to exhaust server resources rather than network resources as in *protocol-based attacks*. Examples of this attack type are HTTP flood, Slowloris, etc. Since these attacks take place in the application layer, they generally do not have a considerable impact on the SDN functions, which mostly run in Layer 4 and below.

In SDN environments, DDoS attacks usually exploit the reactive flow principle of SDN as described in Fig. 2. For protocol-based attacks such as TCP-SYN flood, since the attackers send SYN messages with spoofed IP source addresses and ports, each message is considered as a new flow by the SDN switch. The consequence would be a *packet_in* flood towards the controller. On the other hand, for volume-based attacks such as ICMP or UDP flooding, a large traffic volume could saturate the bandwidth of the forwarding plane and the switch-controller channel as well as the buffer of SDN components.

Figure 3 represents possible vulnerabilities in an SDN network. It is of great interest to evaluate the impact of DDoS attack on SDN, not only qualitatively but also quantitatively. Although the SDN architecture is well-known, its performance is extensively studied in the recent years, we found that the impacts of DDoS attacks

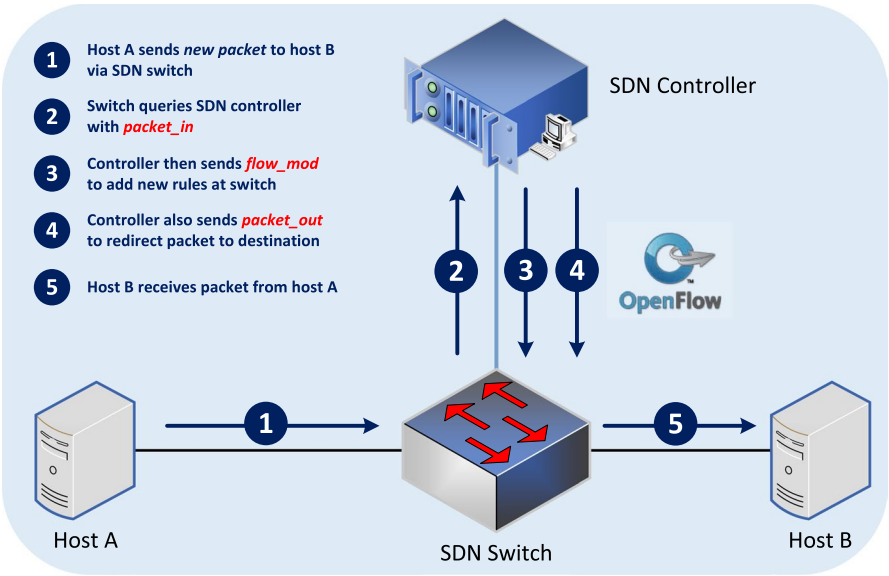


Fig. 2 Data and control plane interactions in SDN

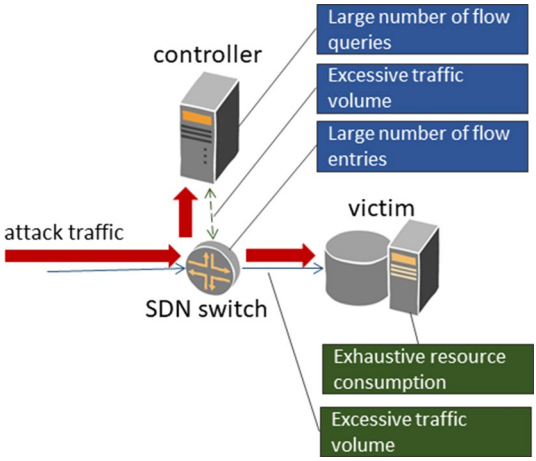


Fig. 3 Summary of vulnerabilities in the SDN architecture

on the SDN architecture, including the forwarding devices and the controller have not been sufficiently investigated yet.

In this paper, we focus on evaluating and benchmarking the performance of SDN components when DDoS occurs. We mainly focus on two types of attacks, namely volume-based attacks and protocol-based attacks. The contributions of this work are as follows:

- A new *test suite* has been developed, which enables real test scenarios other than emulations such as Mininet [5]. The test suite enables quantitative testing and evaluation of SDN components during attacks.
- The *performance* of widely used SDN controllers, switches and controller–switch links is benchmarked in *stress scenarios* where the network components reach their limits. The performance of Ryu [6], POX [7] and Floodlight [8] as well as SDN-enabled switches is extensively evaluated and compared under different DDoS attacks.
- The *behaviours* of SDN components against volume-based and protocol-based attacks is extensively studied. The results of the work show that the SDN architecture is more sensitive to DDoS attacks compared to the traditional distributed network paradigm. By providing comprehensive explanations and analysis of the derived observations, the results provide readers with an in-depth understanding of potential new threats and vulnerabilities introduced by the nature of SDN.

The rest of this paper is organized as follows. Section 2 addresses related work. The development of the proposed testbed architecture is described in Sect. 3. Section 4 addresses and analyzes the impacts of DDoS attacks on SDN network based on real measurement results. The last section is the conclusions and future work.

2 Related Work

This section gives an in-depth review on recent work related to the performance evaluation of SDN in general as well as assessment of SDN performance in case DDoS takes place.

2.1 Recent Research on Security in SDN

Research on security in Software-Defined Networking can be classified into three areas, as identified in Fig. 4.

The first research area covers issues related to *security threats and vulnerabilities introduced by SDN*, which is the scope of this article. As mentioned in the previous section, this research investigates the impacts of DDoS on SDN components, including the controller, the switch as well as the link interconnecting these components. Further discussions will be found in Sects. 2.2 and 2.3.

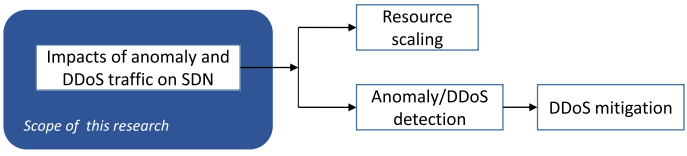


Fig. 4 Research directions related to performance and security in SDN

As anomalies in general or DDoS in particular can cause the saturation of different resources, such as CPU, memory and link capacity at various SDN components, there are two different ways to countermeasure DDoS, namely *resource scaling* and *DDoS detection/mitigation*.

Resource scaling focuses on methods to locate or reallocate network resources to adapt them to network dynamics in anomaly situations. Resource scaling is often useful in some situations. For instance, in case that incoming traffic increases unexpectedly, or a faulty network device sending a large volume of traffic into the network [9–11]. Resource scaling allows avoiding denial of services as a corresponding amount of resources is reserved to serve network requests. However, network resources are costly and not always available.

On the other hand, if anomalies are caused by an attack, there are methods to detect the type of attacks as well as to counteract them in the SDN domain. Houda et. al. [12] present an optimized ML module in SDN that capable of detecting attacks effectively with low computational complexity. Tuan et. al. [13] proposes light-weight ML algorithm based on Local outlier Factor (LoF) to alert abnormal traffic using throughput and flow rates as the features. WisdomSDN [14] can relieve the TCAM space of SDN switch from DNS attack with ML-based detection module and mitigation rules. Another ML approach in SDN-based ISP networks [15] is able to detect TCP or ICMP attack flows based on entropy and mitigate them by imposing block rules. Cochain-SC [16] not only protects intra-domain SDN network from DDoS, but also extends to inter-domain DDoS mitigation with the utilization of Blockchain.

2.2 Performance Issues Related to the SDN Architecture and Components

Regarding performance evaluation of the SDN architecture, current state-of-the-art studies can be categorized into: (1) performance analysis of SDN controllers; and (2) performance analysis of SDN-enabled switches. In the first criteria, Khattak el at. [17] benchmark latency and throughput of two controllers, namely OpenDaylight and Floodlight. Results show that OpenDaylight achieves significantly lower response time than that of Floodlight. However, the paper only addresses performance issues of the controllers in data center context. Also, the behaviours of these controllers under highly loaded traffic were not the focus of the work.

The authors in [18] conduct a comprehensive comparison of more than thirty different controllers in terms of their properties and capabilities such as programming language, architecture, supported platform, etc. Then, nine controllers are analyzed quantitatively with regard to latency, throughput, CPU utilization, etc. Despite thorough results, these controllers are benchmarked in a virtualized environment rather than in a real testbed, which may yield unrealistic outcomes that does not reflect the properties of real SDN networks. Similarly, Mostafavi el at. [19] evaluate a wide variety of SDN controllers' properties as well as Quality of Service parameters in a virtual environment comprising *Mininet* [5] and benchmark tools. The work of Bholebawa et al. [20] also adopts Mininet as an emulation tool for evaluating

performance of two well-known controllers, namely POX and Floodlight over various network topologies. Results show that Floodlight outperforms POX in both round-trip time and throughput. In fact, Mininet is an emulation environment for functional test. We wonder if it could be used for performance test, especially in extreme situations, where the network reaches its performance limits.

Abdullah et al. [21] further investigate the impact of increased request load on the controller in terms of throughput and response delay. Furthermore, the limited number of connected switches that the controller can serve is also identified. Zhao et al. [22] evaluate and compare two groups of controller, the centralized controllers (POX, Ryu, NOX, Floodlight, Beacon) and the orchestrators (ONOS, OpenDaylight) in term of throughput, latency, programming languages and threading analysis.

With regard to the performance of SDN switches, Ngoc et al. [23] introduce and compare two evaluation mechanisms for SDN switches upon processing *flow_mod* messages. While the first is a software-based method, the second one involves a dedicated testing system called Spirent C1. Results show that both mechanisms achieve a sub-millisecond accuracy in terms of processing time. He et al. [24] analyze the impact of control plane on SDN applications. The evaluation scenarios are fast rerouting during link failure and fine-grained traffic engineering in data centers. They also conduct a comprehensive measurement of control plane latency using four different commercial SDN switches. In [25], the authors examine the packet processing dynamics of OpenFlow switches, especially the impact of packet forwarding behaviour of kernel module to meet high performance network and traffic engineering demand. Once again, Mininet is used to evaluate both hardware and software switches.

Bianco et al. [26] focus on the data path and analyze the OpenFlow implementation in Linux based PCs. OpenFlow forwarding mechanism is compared with that of Layer-2 Ethernet switching and Layer-3 IP routing in several performance parameters such as throughput and latency under different loads and traffic patterns. System scalability with diferent forwarding table size and fairness in resource allocation is also evaluated.

Costa el at. [27] examine the performance of some key features on 11 different hardware and software OpenFlow switches. A tesbed is developed using POX and ONOS with packet delay, jitter and packet size as the performance metrics. Kúznia et al. in [28] analyze performance of six hardware switches. The characteristic behaviours of flow table update rates are studied in the work. In [29], capacity and forwarding performance of four HP commercial SDN switches with hardware and software flow tables are investigated.

2.3 Impact of DDoS Attacks on SDN Network

Besides work that investigates the vulnerabilities of the SDN architecture in general, several other research focuses on the particular impacts of DDoS attacks on SDN.

In [30], the authors investigate the impact of DDoS attacks in terms of throughput and load on the controllers in the Software-Defined Internet-of-Vehicles

environment. *Iperf* [31] and *hping3* are used to generate TCP/UDP flows and emulate *ping* attacks in Mininet-Wifi emulator [32]. Sangodoyin et al.'s work in [33] shares some similarities such as simulation tools, yet one parameter being measured is jitter when DoS attack takes place. In [34] the impact of DoS attack on the bandwidth of two different linked hosts (server/client) in SDN networks is examined, where the controllers are POX, RYU, and Opendaylight (ODL). Network performance is tested within Mininet environment by using standard tools such as hping3 and Iperf, etc. Also, controllers' performances against DoS attack under UDP and TCP are assessed. Evaluation parameters are Round Trip Time (RTT), jitter, bandwidth and throughput. In [35] the characteristics of multiple DDoS attack methods and their effects on both data plane and control plane are investigated. Similarly, in [36] Packet Drop Ratio (PDR) and CPU load on both data and control planes are used as the performance metrics to investigate three different controllers (Ryu, ONOS, Floodlight) as the flooding attack rate increases. Mladenov et. al. [37] study the effect of DDoS attack over the southbound channel based on RTT of various topology scenarios, while [38] experiments the impacts of DDoS on switch's flow table and control plane in terms of packet drop rates. The authors in [39] study the effect of slow attack towards SDN-enabled switches by exploiting the limitation of TCAM memory, causing overload to the flow table. Both [40] and [41] theoretically study the effect of DDoS attacks toward three SDN components, which are the controller, the switch and the link between them.

2.4 Discussions and Problem Formulation

As SDN is widely used in different contexts, ranging from 5 G, SDN-WAN to security and virtualization applications in edge-cloud environments and so forth, often networking engineers design their SDN-based solutions by integrating existing products with added software functionalities. Because an SDN solution is more vendor-independent, more flexible and customizable, it is also required that the network with additional components be prototyped and tested carefully in terms of both functional and performance before real deployment. Thus, the following questions are usually raised:

- How an integrated SDN solution based on available and newly developed software and hardware components can be tested, so that major performance vulnerabilities can be found? In other words, how to quantitatively evaluate and benchmark the SDN system to localize bottlenecks and vulnerabilities in the architecture that would cause possible service disruptions?
- How are the performance limits of these SDN components under highly-loaded, stress situations, especially the behaviours of widely used controllers, SDN-enabled switches and the SDN protocols in different DDoS attack scenarios; how can they meet the performance requirements under such specific stress scenarios?

Although the SDN architecture is well-known, its performance has been extensively studied in the recent years, the above survey shows that the impacts of anomaly traffic in general as well as DDoS attack in particular on the SDN architecture have not been sufficiently investigated yet. As discussed in the previous subsections, some previous work evaluates the performance of SDN components based on simulations or such emulation tools as Mininet. By providing a virtualized environment facilitating quick prototyping and testing of SDN networks and functions, emulation tools could be suitable for functional test rather than performance test. As the hardware resources are shared between logical components, the performance of virtualized units, such as virtual links, switches and controllers could be quite different from the real ones. On the other hand, we find that most of the previous work did not focus on stress test of SDN components to investigate their limits, which are very important to understand the behaviours of the network in critical situations. Some other work pays special focus on the impacts of DDoS attacks on SDN. Yet the performance of SDN under attacks is not sufficiently investigated, especially the performance of some widely used SDN controllers and switches in highly loaded situation caused by attacks.

The key features of all aforementioned related work are summarized in Table 1. Among them, Table 2 further discusses the most recent state-of-the-art approaches, addresses some remained issues and gives an overview of the contributions of this research compared to the previous ones. We namely focus on detailed analysis and benchmark of SDN network performance and its major components, including the controller, switch, link in case of DDoS attacks and studies their impacts in stress situations, where the capacities of the components reach their limits. We also benchmark the performance of SDN controllers with additional functionalities compared with the native ones to study the impact of adding new functions on the controller performance.

3 Testbed

In contrast to most other related work, which often use emulation softwares (Table 1) to investigate system performance, this work makes use of a physical testbed environment that allows extensively measuring, benchmarking and analyzing the impacts of DDoS attacks on SDN. Figure 5 describes the detailed testbed configurations comprising four main components, which are: (1) the *Botnet emulator*; (2) the *SDN network*; (3) the *server farm*; and (4) the *data collection and analysis*.

The *Botnet emulator* is used to generate different DDoS attack types just like they could come from botnets in real network scenarios by utilizing software and hardware traffic generators. As for software traffic generators, the testbed makes use of *TCPReplay* [45] and *Bonesi* [46]. Besides that, two commercial Candela hardware traffic generators [47] are also deployed in the testbed. In contrast to software traffic generators, the hardware ones are able to generate a larger volume of traffic, up to 10 Gbps. They can read such real DDoS traffic traces as CAIDA [48] or CIC-DDoS [49] and regenerate them into the testbed. Traffic from different generators

Table 1 Taxonomy of recent work related to benchmarking and performance evaluation of SDN architecture

Related work	Setup	Controller used	Stress test	Analyzed performance parameters			Analyzed SDN components	Category
				Throughput	Latency	Flow table		
[17]	Emulation	ODL, Floodlight	✓	✓	✓		Controller	Performance (Sect. 2.2)
[42]	Testbed	ODL			✓		Controller	
[18]	Emulation	9 Controllers	✓	✓	✓		Controller	
[19]	Emulation	POX, ODL, Ryu, Floodlight, ONOS		✓	✓		Controller	
[20]	Emulation	POX, Floodlight		✓	✓		Controller	
[21]	Emulation	libfluid, ONOS, ODL, POX, Ryu		✓	✓		Controller	
[22]	Emulation	POX, Ryu, NOX, Floodlight, ODL, ONOS, Beacon	✓	✓	✓		Controller	
[23]	Testbed	ODL				✓	Switch	
[24]	Testbed				✓	✓	Switch	
[25]	Emulation	POX, Floodlight			✓	✓	Switch	
[26]	Testbed		✓	✓	✓		Switch	
[27]	Testbed	POX, ONOS			✓	✓	Switch	
[28]	Testbed				✓	✓	Switch	
[29]	Testbed			✓		✓	Switch	
[35]	Emulation	Floodlight		✓			Controller, Switch	
[36]	Emulation	Ryu, ONOS, Floodlight		✓			Controller, Switch	
[37]	Emulation	Floodlight			✓		Link	
[43]	Testbed	POX		✓	✓		Controller, Switch	DDoS (Sect. 2.3)
[38]	Emulation	NOX		✓			Controller, Switch	
[30]	Emulation	Ryu		✓			Controller	
[33]	Emulation	ODL		✓	✓		Controller	
[34]	Emulation	POX, Ryu, ODL		✓	✓		Controller	
[39]	Emulation	Ryu				✓	Link	

Table 1 (continued)

Related work	Setup	Controller used	Stress test	Analyzed performance parameters			Analyzed SDN components	Category
				Throughput	Latency	Flow table		
[41]	N/A						Controller, Switch, Link	
[40]	N/A						Controller, Switch, Link	
Ours	Testbed	POX, Ryu Floodlight	✓	✓	✓	✓	Controller, Switch, Link	Performance, DDoS

Table 2 Comparison with most recent state-of-the-art

References	Descriptions	Open issues	Our contributions
Zhu et al. [18]	Comprehensive qualitative and quantitative analysis of well-known SDN controllers	SDN controllers are solely benchmarked in an emulated environment	A new test suite that enables real-world test scenarios to evaluate and benchmark SDN components in DDoS attacks
Singh et al. [40], Jagdeep et al. [41]	Study the effect of DDoS attacks on SDN architecture and review defense strategies	Only provide qualitative analysis of the impacts	The behaviour of the SDN architecture against DDoS attacks is quantitatively examined by us. New threats and vulnerabilities due to the nature of SDN are discussed: see Sects. 4.2.2, 4.3.1, 4.3.2
Costa et al. [27]	Investigates the performance of hardware and software OpenFlow switches	Functional role of the SDN controller is neglected. Flow rules are pre-installed in the switch	Present behaviour of SDN controllers in case of limited TCAM is investigated
Sun et al. [44]	Identifies race conditions in an OpenFlow network and proposes solution	Only provides theoretical impacts of the race conditions	Illustrate visually the impacts of race conditions on OpenFlow based on testbed evaluations
Kúzniar et al. [28]	The behavioural characteristics of flow table updates are studied in the work	Does not indicate the impact of race conditions	Presenting the performance degradation of flow table due to the race conditions

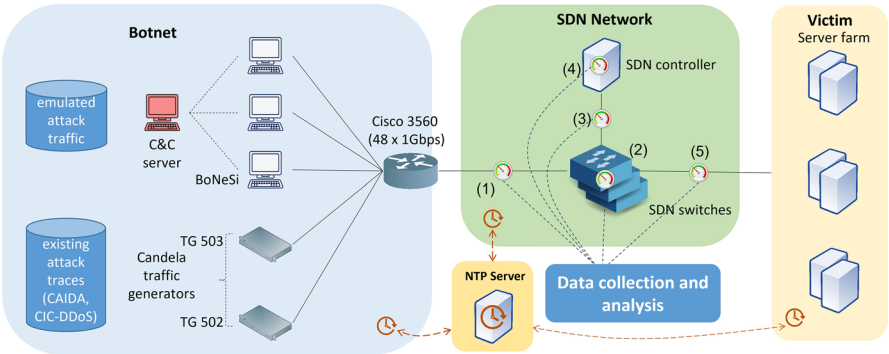


Fig. 5 Testbed architecture

is then cascaded using a Layer-3 Cisco switch to maximize the volume of the traffic sending to the SDN network.

The *SDN network* is the main object of the testbed that is subject to be investigated and benchmarked. In the testbed, SDN components are all implemented on different physical servers to ensure accurate performance evaluations. Physical resources such as CPU, memory and link bandwidth consumed by each component are well isolated. *The SDN Controller* is the network intelligence in the SDN architecture, which manages all operations of the data plane and is a possible single-point-of-failure as previously discussed. Thus, it is worthy to investigate the controller thoroughly in terms of processing capacity, response time as well as different factors affecting its performance. *SDN switch* is the forwarding device in the SDN architecture. Based on the rules imposed by the controller, the switch may process each flow independently by deciding how the switch processes incoming packets. Forwarding rules are stored in the *flow table* in the SDN switch, which is commonly implemented with Ternary Content Addressable Memory (TCAM). The flow table is of interest in this research as TCAM is power-hungry, expensive, and available in limited space or capacity. The SDN switches can be built in hardware or software.

The *victim server farm* in the testbed is used to observe the consequences of the DDoS attacks from the user's perspective as well as to collect attack data going through the SDN domain.

Data collection and analysis collects different information via 5 different measurement points of the network that are numbered in Fig. 5 as (1) incoming traffic to the SDN switch, (2) flow table measurement, (3) OpenFlow traffic between the switch and controller, (4) processed data in the controller, and (5) outgoing traffic from the SDN switch to the server farm. Despite SDN concept has been under research for years, analyzing tools for comprehensive OpenFlow packets inspection are limited. We have to develop our own analyzing tools based on Scapy [50] to extract important information from trace files for detailed investigation. The extracted parameters are new-flow requests per second, bandwidth, number of rules to be installed in the switch per second, flow table size

and flow processing delays. In order to ensure preciseness of the measurements, all components in the testbed have been provided timestamp synchronization with Network Time Protocol (NTP).

4 Impacts of DDoS Attacks on SDN Architecture

The reactive flow installation process in SDN allows dynamic control over the entire network. However, it also exposes vulnerability to DDoS attacks as previously discussed in Fig. 2. This research focuses on *protocol-based* and *volume-based* attacks, as the *application layer attacks* mainly affect Layer 7 in the OSI model and do not have a major impact on the SDN system. Without loss of generality, TCP-SYN flood and ICMP/UDP flood are selected as the major embodiments of the two aforementioned attacks since other attacks belonging to these types could cause the same network behaviours.

4.1 Testbed Setups

The test configurations and parameters are described in Table 3. Real attack traffic traces as well as emulated attacks can be generated by using such software traffic emulators as *TCPReplay* [45] or *BoNeSi* [46], or by the hardware traffic generators deployed in the developed testbed.

In the experiments, the *botnet* performs TCP-SYN and ICMP flood attacks to the victims located in the *server farm*. As the traffic should go through the SDN switch and its controller, we are able to observe the impacts of the attacks on the SDN components. In order to perform stress tests to benchmark the limits of the SDN devices, the attack traffic in terms of *flow numbers* as well as *traffic volume*

Table 3 Testbed configurations and parameters

Controller	POX, Ryu and Floodlight (hosting hardware: 6 Xeon(R) 2.53 GHz, 16 Cores, 32 GB RAM)
SDN Switch	*Software*: Open vSwitch (v2.14.90) (hosting hardware: 6 Xeon(R) 2.67 GHz, 24 Cores, 64) *Hardware*: HPE Aruba 2920 24 G - J9726A
Victim	High performance server (6 Xeon(R) 2.67 GHz, 24 Cores, 64GB RAM)
Traffic generator	BoNeSi, TCPReplay (hosting hardware: Xeon(R) 3.70 GHz, 12 Cores, 16GB RAM)
OpenFlow	OpenFlow 1.0 and OpenFlow 1.3
Type of attack	
SYN flood	*Rate*: 5, 000 fps in 10 sec for all controllers; 10, 000 fps, 15, 000 fps, 20, 000 fps in 10 sec for Floodlight; *Number of bots*: 50, 000
ICMP flood	*Bandwidth*: 56 Mbps - 900 Mbps *Number of bots*: 50, 000 *Payload length*: 1, 400 bytes
Measurement parameters	*Controller*: number of incoming requests (*packet_in*), outgoing rules (*flow_mod*), processing latency *Switch*: incoming traffic volume, throughput, flow table size, latency

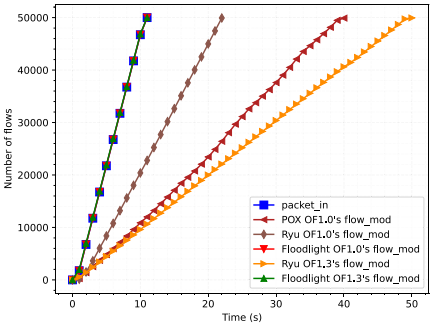
increases gradually to 20, 000 fps for TCP-SYN and packet length of 1400 bytes each for ICMP flood, respectively. A flow in all experiments is defined as a tuple of *{IP source address, IP destination address, source port, destination port}*.

The following SDN controllers and switches are under investigation:

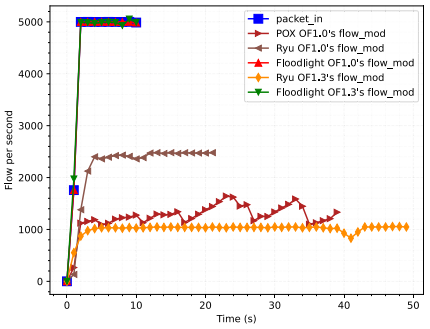
- *POX* [7]: POX is a well-known Python-based open source controller. Besides NOX [51], POX is considered as one of the earliest SDN controllers developed and is used for faster development and prototyping of new network applications. It is pre-installed in several common SDN-based network development tools, such as Mininet.
- *Ryu* [6]: is also a Python-based open source controller widely used by the research and development communities recently. It supports well defined APIs and Python mathematical libraries that can be used easily to create new network management and control applications, especially machine learning-based functions.
- *Floodlight* [8]: is another open source controller based on Java. Floodlight's architecture can be advantageous for developers because it offers the ability to easily adapt software and develop applications, including REST API. With an extensible Java development environment, and enterprise-grade core engine, Floodlight is an easy to use and robust SDN controller. Floodlight is integrated in several open source and commercial products extensively used in the networking industry, including OpenStack.
- *Open vSwitch* [52]: a renown software-based SDN switch designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols. In cloud computing, Open vSwitch (OVS) is an important component allowing network functions to be virtualized to create Service Function Chain (SFC). OpenStack [53] is running Open vSwitch to provide virtualized network services.
- *Aruba 2920* [54]: a layer-3 commercial hardware access switch made by Aruba, an HP Enterprise company. Aruba 2920 is an SDN-enabled switch running OpenFlow Protocol 1.0 and 1.3 with 24 ports, including four 10Gbps-ports. Although Aruba can store up to 2 048 IPv4 entries in its routing table, no data have been provided on the number of OpenFlow entries that its flow table can accommodate. In this research, we choose Aruba as a reference hardware switch used to compare its performance with such widely used software solutions as Open vSwitch.

4.2 Impacts of DDoS on Controllers

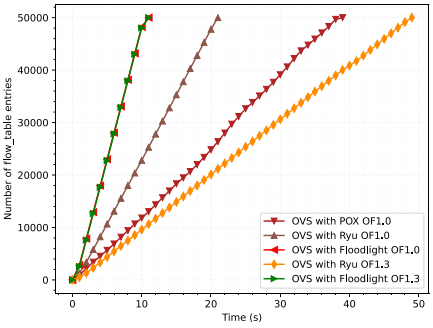
This section investigates the impacts of DDoS attacks on the controller. In order to evaluate the performance of the controllers, the testbed make use of OVS as the SDN switch in all test scenarios.



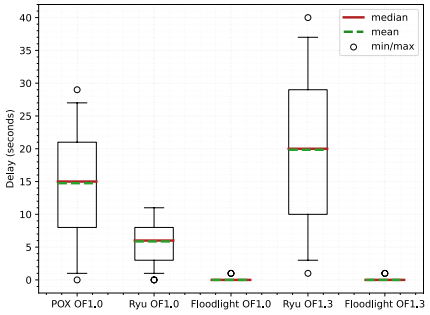
(a) Cumulative number of incoming and outgoing flows at controller



(b) Incoming/outgoing flows per second



(c) Flow table sizes



(d) Delay statistics

Fig. 6 Performance of controllers under 5000 TCP-SYN incoming flows per second

4.2.1 Impacts of Protocol-Based Attacks on Controllers

Firstly, a TCP-SYN attack with 5000 fps (or equivalent to 3.04 Mbps) in 10 seconds is generated into the network. As the spoofed IP addresses and ports are unique, each TCP SYN message is considered by SDN as a new flow. These new flows consume both resources of SDN-enabled switch for flow table look-ups and controller for handling *packet_in* messages. Figure 6a show the cumulative number of incoming flows, which is the *packet_in* messages, and the number of outgoing flow rules carried in *flow_mod*s measured at POX, Ryu and Floodlight. The slopes of the curves in Fig. 6a represent the incoming rate as well as the processing rates of the controllers. If the incoming and outgoing slopes are coincident, the controller is capable of processing all incoming flow queries, otherwise the controllers are overloaded. This can be depicted more clearly in Fig. 6b, which represents the processing rates of each controller compared to the incoming queries. As can be seen, while Floodlight can well accommodate the number of queries at 5000 fps, POX and Ryu are already overloaded and only able to process up to 1200 fps and 2400 fps, respectively.

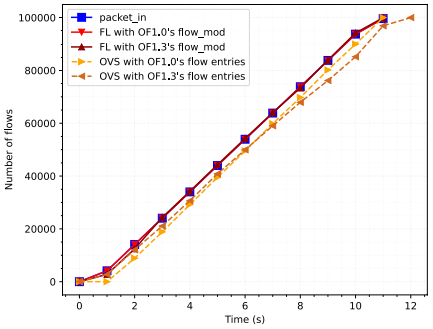
On the other hand, Ryu running OpenFlow 1.3 (OF1.3) performs worse than that running OpenFlow 1.0 (OF1.0), which is only able to handle around 1000 fps. Meanwhile, Floodlight with OpenFlow 1.3 can still adapt with the same throughput as OpenFlow 1.0. The evolution of OpenFlow protocol from 1.0 to 1.3 have witnessed the introduction of new features such as multiple tables, OXM match to extend matching flexibility, role change mechanism for scalability, meters for QoS support, table-miss entry and various new OpenFlow messages [55]. These changes add more complexity in the SDN controller, which presumably reduces the processing performance as shown (Fig. 6).

We further explore the impact of TCP-SYN flood on the SDN switch. Figure 6c shows the cumulative number of entries stored in the switch. As shown, the number of entries increases proportionally with the number of rules in *flow_mod* messages that the switch receives from the controller as represented in Fig. 6a. This implies that all rules can be installed and the bottleneck in this case is the controller.

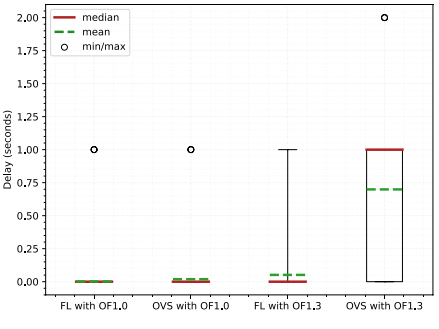
The box plots (Fig. 6d) present the processing latency statistics in the controller, calculated as the distribution of the difference between *flow_mod*'s departure and *packet_in*'s arrival times. In these plots, the caps on both ends of the whiskers occupy for 5- and 95-percentile delays and the outermost fliers represent the minimum or maximum delay values, also the first (Q1/25-percentile) and third (Q3/75-percentile) quartiles are displayed by both ends of the central box. As can be seen, due to controller overload, the average processing delays of POX, Ryu OF1.0, Ryu OF1.3 are 15, 6 and 20 seconds, respectively, while Floodlight OF1.0 and OF1.3 experience no delay at all as Floodlight can stand well at 5000 fps.

To investigate the processing limit of Floodlight and OVS, we further increase the attack rates to 10, 000 fps, 15, 000 fps and 20, 000 fps, which are equivalent to 6.08 Mbps, 9.12 Mbps and 12.16 Mbps, respectively. We combine the flow table size and cumulative number of incoming *packet_in* and outgoing *flow_mod* (Fig. 7a, c, e) in the same figure in each case to easily compare and evaluate the performance of the switch (see Sect. 4.3) and the controller. The box plots in Fig. 7b, d, f represent the processing delay of Floodlight and its corresponding data packet latency passing through the OVS, measured by the difference between the packet's departure and arrival time.

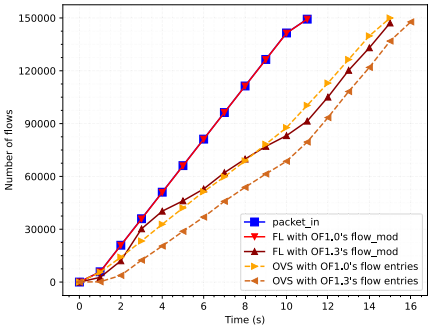
All in all, we observe that TCP SYN attack is malicious to the controller, although the attack traffic volume is minimal. At 10, 000 fps as shown in Fig. 7a, Floodlight with OpenFlow 1.0 can accommodate well to the volume of arrival queries, as the number of outgoing rules (represents as OF1.0 flow_mod in the figures) matches the number of incoming *packet_in* requests. Accordingly, the data packet latency at OVS (OVS with OF1.0) is very low as shown in Fig. 7b. On the other hand, the performance of Floodlight running OpenFlow 1.3 starts to degrade at the end of the test (Fig. 7a), which reflects the fluctuation of processing delay in Fig. 7b (FL with OF1.3). Furthermore, the packet delay at the switch get worse compared to the controller processing delay with an increased average packet delay at around 0.72 sec as shown in Fig. 7b (OVS with OF1.3). Thus, it is while-worthy to note when the attack rate reaches a certain level, OVS also contributes to service degradation.



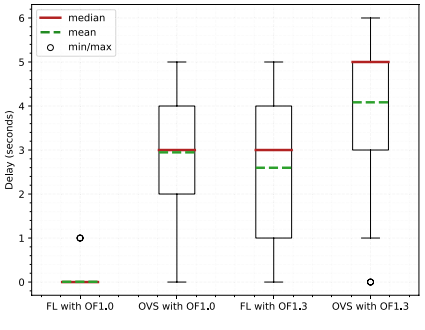
(a) Cumulative number of incoming/outgoing flows (10,000 fps)



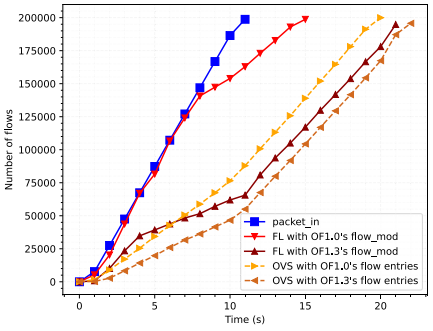
(b) Latency in controller and OVS (10,000 incoming fps)



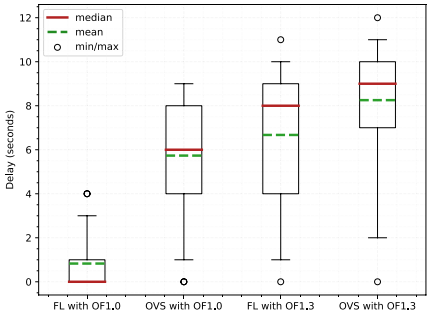
(c) Cumulative number of incoming/outgoing flows (15,000 fps)



(d) Latency in controller and OVS (15,000 incoming fps)



(e) Cumulative number of incoming/outgoing flows (20,000 fps)



(f) Latency in controller and OVS (20,000 incoming fps)

Fig. 7 Floodlight performance under TCP-SYN attack at at 10, 000, 15, 000 and 20, 000 fps

The reason of the increased packet delay at SDN switch will be discussed later on in Sect. 4.3.

The same phenomenon is more visible when the attack rate increases. At 15, 000 fps as shown in Fig. 7c, d, Floodlight with OpenFlow 1.0 can still perform well, while Floodlight with OpenFlow 1.3 degrades considerably. On the other hand, the

Table 4 Peak resource utilization of SDN controllers

Controller	POX	Ryu	Floodlight	
Incoming traffic (fps)	5000	5000	5000	20, 000
Resources				
CPU	100%	100%	212%	470%
Memory	0.1%	0.1%	2.4%	3.7%

curves representing the outgoing *flow_mod*s and the number of table entries are not coincident, indicating that OVS cannot handle a very large number of incoming *flow_mod* messages. The consequence is the increased packet delay in the data plane (Fig. 7d). In the last test case (Fig. 7e, f), even Floodlight with OpenFlow 1.0 cannot withstand such a large amount of new 20, 000 flow requests per second.

Table 4 indicates the peak resource utilization of the SDN controllers in the aforementioned experimental scenarios. Since POX and Ryu are single-threaded controllers, under highly loaded situations both their peaks can reach 100% CPU usage (1-core CPU utilization as in Table 3). In contrast, Floodlight witnesses its resource usage increases in proportion to the traffic load. Its CPU utilization at 5000 and 20, 000 fps are 212% (3-core CPU utilization) and 470% (5-core CPU utilzation), respectively. The results show that Floodlight is more scalable and robust compared to other controllers thanks to its multi-threaded implementation.

Table 5 shows further benchmarks of the three controllers under various test environments, in which the performance of POX, Ryu and Floodlight are analyzed. We compare the controller capacity in terms of the number of flows processed per millisecond under emulated environments, such as Cbench or Mininet with that under real testbed. The results further elaborate our arguments previously stated in Sect. 2.4 that emulation, simulation and testbed may yield different results due to the differences in hardware, software and implementing environments. While Cbench is considered as a traffic generator capable of sending *packet_in* messages as fast as possible [18, 22], actual switch typically contains a smaller sending buffer and lower sending rate. Thus there exists the different order of magnitude in terms of *responses per ms* between the two settings [56]. This work has set up a complete SDN architecture that can reflect actual results in real-world scenarios. One another key takeaway from this comparison is that, despite of the differences in performance under various test environments, all results show that Floodlight outperforms other two controllers in terms of processing capacity and POX is the least performed controller.

Table 5 Controller performance benchmark (processed flows per *ms*)

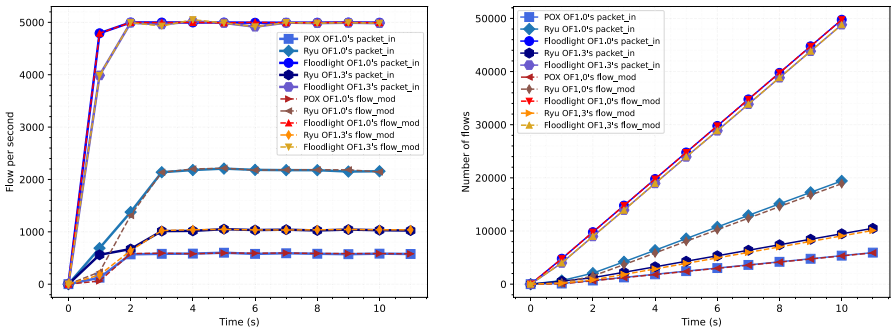
	Zhao [22]	Jawahara [56]	Zhu [18]	Our results
POX	105	0.15	150	1.2
Ryu	106	N/A	200	2.4
Floodlight	670	N/A	420	19
Environment	Cbench	Mininet	Cbench	Testbed

In summary, under protocol-based attacks, if the attack rates are large enough, both controller and SDN switches suffer. The controller is overloaded by processing a large number of new flow requests. On the other hand, the switch cannot process a large number of *flow_mod* messages sent by the controller, leading to the delay in forwarding data packets from the inputs to its outputs. We further analyze the phenomenon of switch performance degradation in Sect. 4.3.

4.2.2 Impacts of Volume-Based Attacks on Controller

In the next test set, ICMP flood is used to investigate the impact of volume-based attacks on SDN controllers. The main purpose of a volume-based attack is to saturate the Internet connection of the victim by sending a large volume of traffic to the victim host. Thus, in conventional destination-based forwarding paradigm, ICMP attacks as a typical volumetric attack may have negative impacts only on the data paths but not on the control plane, independent of the number of new flows carrying ICMP traffic. In SDN, an ICMP packet is classified by the controller with the same flow tuple as TCP packets, but can be configured with larger payload size, hence presumably causing different behaviour to SDN controllers. To verify this, an attack scenario with 5000 ICMP pps, which is equivalent to 56 Mbps in 10 seconds has been set up. Similar to the TCP SYN attack scenario described previously in Fig. 6b, each ICMP packet has an unique flow tuple but with 1, 400 bytes payload.

As can be seen in Fig. 8a, the incoming *packet_in* rates to Ryu OF1.0, POX OF1.0 and Ryu OF1.3 are approximately 2000, 600 and 1000 fps, respectively, which is remarkably lower than the arriving ICMP traffic rate of 5000 fps at the input of the OVS. On the other hand, in case of TCP SYN (Fig. 6b), the same number of 5000 TCP flows requests per second well arrived at the controllers. As further shown in Fig. 8b, this results in *packet_in*s loss at the OVS as only a fraction of the total 50, 000 new flow requests can arrive in Ryu and POX. Contrarily, the incoming *packet_in* rate of Floodlight is matched with the arrival rate of new ICMP flows in the system.



(a) Incoming/outgoing flows per second

(b) Cumulative number of flows

Fig. 8 Impacts of ICMP flood on SDN controllers

After thoroughly investigation, we found out that this phenomenon is due to TCP flow control [57] trying to adapt its sending rate to the capacity of the controller. The OpenFlow protocol makes use of secure channel layered on top of TCP [58] to exchange such messages as *packet_in*. The TCP flow control mechanism use the *Receive Window* (*RWND*) to ensure that the sender does not overwhelm the receiver by sending more packets than it can handle [59]. On the other hand, since some SDN-enabled switches such as OVS and Aruba do not support internal buffering, the switch must send full ICMP packets encapsulated in *packet_in* to the controller as a part of the new flow request.

Fig. 9 presents a TCP Window Scaling graph [60] which illustrates how well the controller as the receiver can handle the received data. *RWND* indicates the size of receive buffer, while *Bytes in flight* specifies the amount of unacknowledged data the sender has transmitted. The *Bytes in flight* will always be less than or equal to the *RWND* [60]. As can be seen, POX and Ryu's RWND constantly oscillated and dropped to 0 (*TCP ZeroWindow*) showing that these controllers were not able to handle the incoming traffic fast enough, thus the buffer got filled up, leading to *packet_in* loss at the OVS. In the case of Floodlight, the flat-line of *RWND* signifies a considerable processing capability of this controller as expected. These observations imply that volume-based attacks in SDN are more malicious than in the conventional networking paradigm as they not only saturate the data paths but also have a direct negative impact on the controller. Moreover, volume-based DDoS can even be more malicious than protocol-based attacks in SDN. The attacker, for instance, can use a large number of botnets to send big ICMP/UDP packets to a random IP address located in the destination SDN network. Instead of attacking the destination, this will cause the buffer and computing facility of the SDN controller to be exhausted, thus affecting the whole network. This negative impact takes effect even in case the attacking ICMP traffic is in low rate, e.g., 56 Mbps in this case, and does not necessarily saturate the link bandwidh.

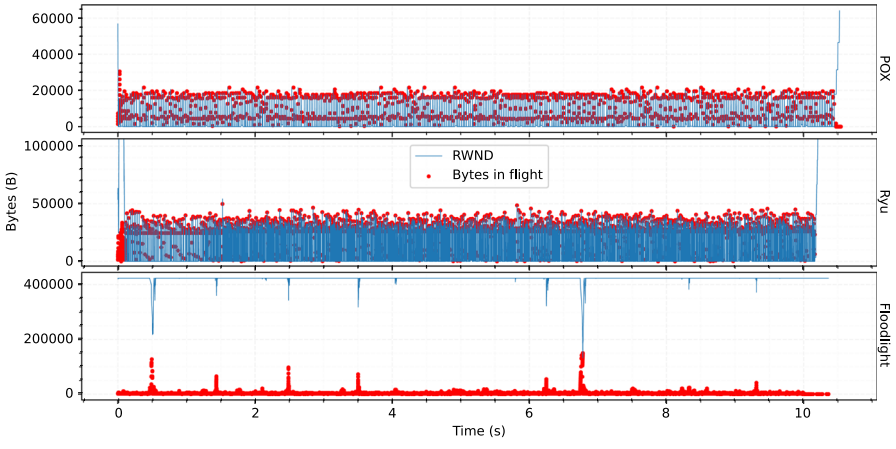


Fig. 9 TCP Received Window Size of SDN Controllers during ICMP flood

The evaluation results in the sections above have shown that Floodlight outperforms POX and Ryu. This is because Floodlight is Java-based controller that supports multi-threading, which can undoubtedly achieve higher performance. Meanwhile, POX and Ryu utilized Python, a scripting language that only well support single-thread, thus limiting its ability to scale the performance. Although the need of designing high performance controller is vital as a logically centralized controller is responsible to manage the entire network, Floodlight and other high performance controllers (e.g. NOX, ONOS, Beacon, etc.) are rigid to implement Machine Learning-based applications. In contrast, Python-based controllers (POX and Ryu) are more flexible to such methods as they facilitate Machine Learning (ML) frameworks and libraries, which offers comprehensive and effective mathematical functions. In fact, many researchers, who developed Machine Learning-based defense methodology against DDoS attacks in SDN context, utilize POX and Ryu [15, 40, 41].

4.3 Impacts DDoS on SDN-Enabled Switch

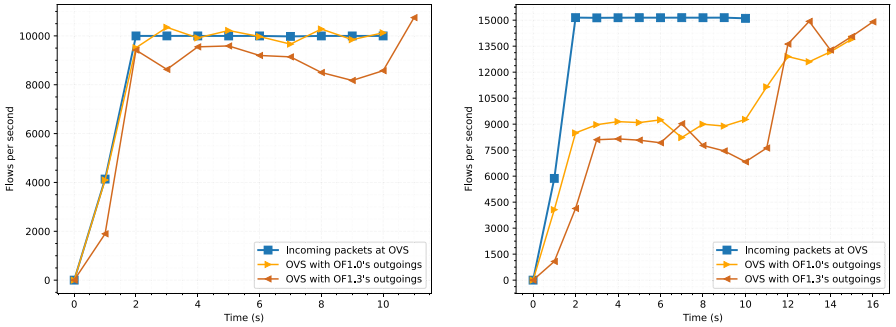
This section investigates the impacts of DDoS attacks on the SDN-enabled switchs, including software and hardware ones.

4.3.1 Flow Table Lookups vs. Updating Flow Table

In this section, we would like to take a more insight analysis of the switch performance. Figure 7 already shows that OVS incurs considerable packet delay. It is because the processing of *flow_mod* messages to add new entries in the flow table cannot catch up with the *flow_mod*s' arrival rate. This section is going to explain this argument in more details. Figure 7c and e show that adding new flow entries in the OVS flow table is more costly than creating new rules in Floodlight as the curve representing the cumulative number of sending *flow_mod* messages is steeper than the curve of cumulative entries adding in the flow table. We observe that in a physical machine with 6 Xeon, 2, 67 GHz, 24 cores (see Table 3), OVS has reached its processing limit of roughly 10, 000 rules per second. If the arrival rate of *flow_mod*s is higher than this threshold, OVS starts showing its performance falling behind the controller.

This performance degradation at OVS is caused by the *race condition* within the switch. This issue has firstly been addressed by Sun et al. [44] but was not clearly explained. In this work we can analyze it in more details by using the new developed testbed. The race condition happens when the *flow_mod/packet_out* messages arriving from the controller and the incoming data packets contend for accessing the flow table concurrently. The former ones are for updating flow table entries while the latter ones access the table for lookup-operations. OVS has implemented *ofproto_mutex* for locking mechanism to handle this problem. However, it will eventually degrade the processing performance if the number of threads trying to access flow table are sufficiently high.

For a clearer demonstration, Fig. 10 depicts the incoming versus outgoing rates of TCP-SYN packets in OVS. In the experiments, the TCP-SYN flood lasts for 10



(a) Incoming/outgoings traffic at OVS with new flow requests at 10,000 fps

(b) Incoming/outgoings traffic at OVS with new flow requests at 15,000 fps

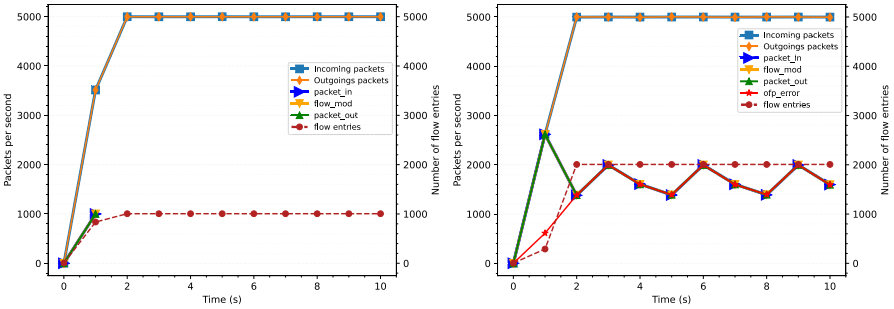
Fig. 10 OVS with Floodlight performance under TCP-SYN attack (corresponding with scenarios in Fig. 7a, c)

sec with the arrival rate of 10, 000 pps and 15, 000 pps, equivalent to 10, 000 fps and 15, 000 fps of *packet_in*s, respectively. The controller used in this experiment is Floodlight. In both cases, during the attack the outgoing packet rates are lower than the incoming rates. Once the attack stops, the outgoing packet rates increase substantially. This can be explained as follows. During the attack, incoming TCP-SYN packets cause the switch to initiate new flow requests to the controller. The controller, in turn, install new rules by sending out *flow_mod*/*packet_out* messages that cause the aforementioned race condition between the incoming TCP packets and *flow_mod*/*packet_out* messages at the switch. At the end of the attack, as there is no more incoming TCP packets contending for accessing the flow table, the race condition is resolved. The switch continues to forward unsent data packets to the destinations with increased outgoing rates.

4.3.2 Impact of DDoS on Hardware Switch with Limited Hardware Resources

Apart from software-based switches such as the OVS, hardware SDN-enabled switches often utilize Ternary Content Addressable Memory (TCAM) to implement the flow tables. TCAM is resource expensive and power hungry as pointed out in [39–41]. Therefore, most current commercial SDN-enabled switches have a limited TCAM capacity and can contain only a limited number of rules, typically from 750 to 4500 rules [28].

In order to study the effect of the small flow table size on performance, we firstly emulate the behaviour of the hardware switch by limiting the flow table capacity of the OVS to 2000 entries. Figure 11 illustrates this effect. We perform ICMP flood at the traffic rate of 5000 pps, or 56 Mbps in 10 sec with the bot number of 1000 and 3000, corresponding to 1000 and 3000 flows in total for each case. Thus, each flow in the above cases contains 50 and 17 ICMP packets, respectively. The controller used in this test is Floodlight. Figure 11a describes the normal case when the number of 1000 new flow requests (from 1000 bots) is less than maximum flow table capacity (2000 entries). As can be seen, the controller stopped receiving *packet_in*s after the first second. After



(a) ICMP attacks with 1,000 bots (b) ICMP attacks with 3,000 bots

Fig. 11 Network behaviours in case of limited TCAM space (2, 000 entries) under ICMP attack

all 1000 rules acquired from *flow_mod* messages have been installed in the flow table, the incoming traffic is forwarded straight to the output based on flow table lookups. In contrast to the first case, Fig. 11b shows the issues in SDN as the number of flows (i.e., 3000 bots) is greater than the maximum flow table capacity (2000 entries). Because there is insufficient space in its TCAM memory for installing new rules, the switch drops *flow_mod* packets (*packet_out*s are still processed normally). It also notifies the controller that the TCAM memory is full by *ofp_error* messages with the rejected *flow_mod* packets included in the payload. From this point, for every new incoming flow the switch: (1) continuously queries the controller with *packet_in* messages; (2) forwards packets to output port by *packet_out* messages; and (3) drops *flow_mod* events then informs the controller with *ofp_error*s. In our case, the controller continuously receives *packet_in*s with the rate of roughly 1000 fps. Other incoming packets that are matched with the entries in the flow table can be directly forwarded without querying the controller.

From the above experiments, it is observed that overflowed TCAM memory has made the controller to be continuously functioning as there is no room left for new rules, which could saturate its computing resource. Also the switch–controller channel is always occupied by OpenFlow traffic that could lead to congestion. When the buffer of the switch is full, *packet_in* with the whole packet payload will be sent instead of only packet header. This action puts more stresses to the controller–switch communication channel.

As discussed above, TCAM resources are expensive, and typical hardware switches can only contain up to 4500 rules [28]. We want to see how is the impact of DDoS on a hardware switch with limited TCAM resource by using *HP Aruba 2920*, a commercial OpenFlow hardware switch. Table 6 shows the maximum TCAM capacity of HP-2920 reported in various research, which can be around 500 entries [29, 61]. On the other hand, we have configured HP-2920 to dedicate 100% hardware resources [62]

Table 6 Reported hardware TCAM capacity of HP-2920

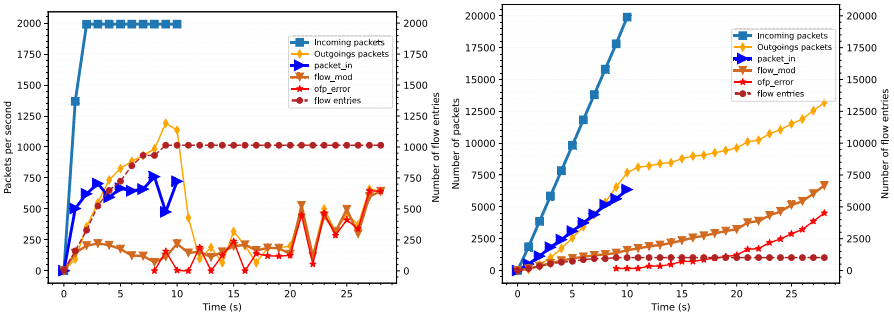
	Piotr [29]	Bauer [61]	Our setup
Flow table size	460	500	1014

for OpenFlow services and observed the maximal TCAM capacity of 1014 entries. As the focus of some previous work is rather on the forwarding performance of hardware devices, the functional role of the SDN controller is often neglected and forwarding rules are pre-installed in the switch [27, 29]. By contrast, in this work Aruba behavior in reactive flow insertion is investigated by conducting ICMP attack with the rate of 2000 pps sent from 2000 bots. Floodlight with OpenFlow 1.3 is used in the test as the high-performance controller.

As shown in Fig. 12, the hardware switch behaves generally the same as previously analyzed in Fig. 11b. However, its performance is even worse. There is a large amount of packets drop when comparing the outgoing packets with the incoming ones (Fig. 12b). In fact, there are 2, 000 new flows arriving in the switch in the very first seconds. However, Aruba can only send roughly 600 fps to the controller in *packet_in* messages. Moreover, measurement results in Fig. 12 shown that the *flow_mod* rate is below *packet_in* rate as Aruba restricts the *TCP Receive Window* of the controller southbound interface due to its limited processing capability as previously discussed in Sect. 4.2.2.

As can be seen in Fig. 12a, Aruba keeps sending out all backlogged *packet_in*s until the 10th second. At this time, only roughly 7, 500 packets are sent out of the switch compared to 20, 000 incoming packets (Fig. 12b). After the 10th second, Floodlight continues to send the remaining buffered *flow_mod* messages to Aruba, thus the total number of outgoing packets eventually reaches roughly 13, 500. *ofp_error*s were sent to indicate that the TCAM has reached its maximum capacity of 1, 014 entries.

Theoretically, with such a capable controller as Floodlight, the *flow_mod* rate must be in line with the *packet_in* rate (Fig. 6). However, that is not the case as shown in Fig. 12. This is once again due to TCP Flow Control explained in Sect. 4.2.2. With its limited performance [27], Aruba 2920 cannot catch up with Floodlight. Hence the hardware switch regularly advertises empty *RWND* to the controller, and Floodlight will gradually releases *flow_mod*s to the switch based on the updating *RWND* value accordingly.



(a) ICMP attacks with 2,000 bots

(b) Cumulative number of packets

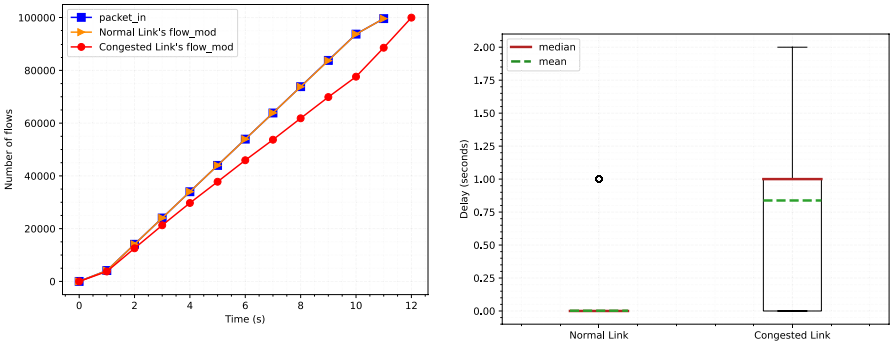
Fig. 12 HP Aruba 2920 Switch behaviour under ICMP attacks

These above observations show that hardware OpenFlow switches may outperform software switches in terms of throughput in normal operations. However, in critical situations their performance could be limited due to scarce hardware resources. On the other hand, high performance commercial TCAM-based switch usually comes with high investment expenses compared to open source, software ones. Recently, OVS with DPDK acceleration [63, 27] is often considered a high performance solution with low cost and no vendor lock-in.

4.4 Impacts of Volume-Based DDoS Attacks on Controller-Switch Link

The controller–switch link, also known as the *southbound interface*, provides a secure channel between switches and the controller via OpenFlow protocol. It is important that the controller should remain connected with the switches all the time to provide flow-based rules and manage the entire network. Volume-based DDoS traffic could constantly occupy the link between switches and the controller, which may create bottleneck and make legitimate packets difficult to arrive on time. Furthermore, this can potentially cause the controller to become unavailable to underlying switches and network hosts if the strength of DDoS attacks is sufficiently large.

In order to study the impact of DDoS on the OpenFlow channel, we intentionally consume 1-Gbps bandwidth of the switch–controller physical link by sending ICMP flood at around $80,3 \times 10^3$ pps, each packet with 1400 bytes. Concurrently, TCP sessions of legitimate users try to set up connections by sending several TCP SYN messages to a server located in the SDN network. Since the network considers these messages as new incoming flows, the switch send *packet_in* messages to the controller on the switch–controller link. We measure the latency calculated as the difference between the departure time of *packet_in*s from the switch and the arrival time of *flow_mod*s. Figure 13a shows that the rates of receiving *flow_mod* packets on congested link fall behind that under the normal case. This can also be seen more clearly in Fig. 13b illustrating the distribution of latency. With a more complex topology in the data plane, there will be amplification of OpenFlow messages from



(a) Floodlight OF1.0's *flow_mod* behaviour under congested link

(b) Delay measurements

Fig. 13 Effect of congested controller–switch link on latency in data plane

multiple switches to a centralized controller, the results could be worse. Therefore, this issue impacts on the throughput performance of OpenFlow messages and can degrade Quality of Service of the entire network.

5 Remarks and Conclusions

Several remarks can be drawn from this research, namely:

- While DDoS attacks have an impact mostly on the data plane of the conventional networking paradigm, they are more malicious to SDN as they may affect not only the data plane or end hosts, but also on the control plane of the network. This vulnerability is due to the centralized nature of the SDN control plane and its per-flow processing paradigm, while traditional networks utilize distributed control and destination-based packet processing that is more robust to DDoS.
- The vulnerabilities in SDN can be at the controller, at the switch, or the link interconnecting them. More specifically, both protocol-based and volume-based attacks can burden the controller with high load of flow requests if the number or traffic volume of incoming flows is large enough. We also point out that volume-based attacks can have a great negative impact on the controller even at low rate (Sect. 4.2). On the other hand, the processing capability of *flow_mod*s to add new entries in the flow table may hinder switch's normal forwarding operations (Sect. 4.3). This can happen when the number of *flow_mod*s is very large. Also, the size of the flow table is an important performance factor of the switch, which may in turn impacts on the performance of the controller later (Sect. 4.3). Finally, congested links between the switches and the controller degrade the overall network performance since it prevents normal communication between the SDN data plane and its control plane (Sect. 4.4).

In this paper, we provide an in-depth analysis of the performance of SDN networks and its components in critical situations, when DDoS attacks take place. A testbed architecture has been proposed that allows testing, bench-marking the performance of different components in the SDN architecture. The testbed can be used for stress testing and studying the performance limit of each component in the architecture. Results in the paper allow addressing the bottle necks of SDN in extreme conditions, when the network is overloaded. It is shown that besides its advantages, there are several vulnerabilities in the SDN components that require special attentions when designing and operating SDN networks.

The research is the basis for further development of more scalable and reliable softwarized network architecture and for providing methods to protect network components in critical situations.

Author Contributions All authors contributed equally to this work

Funding No funding

Data Availibility All of data and materials are owned by the authors and/or no permissions are required

Code Availibility All of code availability is owned by the authors and/or no permissions are required.

Declarations

Conflict of interest I declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical Approval Not applicable

Consent to Participate Not applicable

Consent for Publication Not applicable

References

1. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
2. Foundation, O.N.: Software-defined networking: the new norm for networks. ONF White Paper **2**(2–6), 11 (2012)
3. Malik, M., Singh, Y.: A review: DoS and DDoS attacks. Int. J. Comput. Sci. Mob. Computing **4**(6), 260–265 (2015)
4. Mahjabin, T., Xiao, Y., Sun, G., Jiang, W.: A survey of distributed denial-of-service attack, prevention, and mitigation techniques. Int. J. Distrib. Sens. Netw. **13**(12), 1550147717741463 (2017)
5. Mininet. http://mininet.org/. Accessed 25 March 2023
6. Ryu. https://ryu-sdn.org/. Accessed 25 March 2023
7. POX. https://noxrepo.github.io/pox-doc/html/. Accessed 25 March 2023
8. Floodlight. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview. Accessed 25 March 2023
9. Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., Kompella, R.R.: ElastiCon; An Elastic Distributed SDN Controller. In: 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), pp. 17–27. IEEE (2014)
10. Macedo, R., de Castro, R., Santos, A., Ghamri-Doudane, Y., Nogueira, M.: Self-organized SDN Controller Cluster Conformations Against DDoS Attacks Effects. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2016)
11. Wang, Y., Hu, T., Tang, G., Xie, J., Lu, J.: SGS: safe-guard scheme for protecting control plane against ddos attacks in software-defined networking. IEEE Access **7**, 34699–34710 (2019)
12. El Houda, Z.A., Hafid, A.S., Khoukhi, L.: A novel machine learning framework for advanced attack detection using sdn. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2021)
13. Tuan, N.N., Nghia, N.D., Hung, P.H., Tuyen, D.K., Hieu, N.M., Hung, N.T., Thanh, N.H.: An Abnormal Network Traffic Detection Scheme Using Local Outlier Factor in SDN. In: 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), pp. 141–146. IEEE (2021)
14. Abou El Houda, Z., Khoukhi, L., Hafid, A.S.: Bringing intelligence to software defined networks: mitigating ddos attacks. IEEE Trans. Netw. Serv. Manag. **17**(4), 2523–2535 (2020)
15. Tuan, N.N., Hung, P.H., Nghia, N.D., Tho, N.V., Trung, P.V., Thanh, N.H.: A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN. Electronics (Networks Section) **9**(3), 413 (2020)
16. Abou El Houda, Z., Hafid, A.S., Khoukhi, L.: Cochain-sc,: an intra-and inter-domain ddos mitigation scheme based on blockchain using sdn and smart contract. IEEE Access **7**, 98893–98907 (2019)

17. Khattak, Z.K., Awais, M., Iqbal, A.: Performance Evaluation of OpenDaylight SDN Controller. In: 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 671–676. IEEE (2014)
18. Zhu, L., Karim, M.M., Sharif, K., Xu, C., Li, F., Du, X., Guizani, M.: SDN controllers: a comprehensive analysis and performance evaluation study. ACM Computing Surveys (CSUR) **53**(6), 1–40 (2020)
19. Mostafavi, S., Hakami, V., Paydar, F.: Performance evaluation of software-defined networking controllers: a comparative study. Comput. Knowl. Eng. **2**(2), 63–73 (2020)
20. Bholebawa, I.Z., Dalal, U.D.: Performance analysis of SDN/openflow controllers: POX versus floodlight. Wirel. Pers. Commun. **98**(2), 1679–1699 (2018)
21. Abdullah, M.Z., Al-Awad, N.A., Hussein, F.W.: Performance evaluation and comparison of software defined networks controllers. Int. J. Sci. Eng. Sci. **2**(11), 45–50 (2018)
22. Zhao, Y., Iannone, L., Riguidel, M.: On The Performance of SDN Controllers: A Reality Check. In: 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), pp. 79–85. IEEE (2015)
23. Nguyen-Ngoc, A., Lange, S., Gebert, S., Zinner, T., Tran-Gia, P., Jarschel, M.: Performance Evaluation Mechanisms for Flowmod Message Processing in Openflow Switches. In: 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), pp. 40–45. IEEE (2016)
24. He, K., Khalid, J., Gember-Jacobson, A., Das, S., Prakash, C., Akella, A., Li, L.E., Thottan, M.: Measuring Control Plane Latency in SDN-enabled Switches. In: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, pp. 1–6 (2015)
25. Aliyu, A.L., Bull, P., Abdallah, A.: Performance Implication and Analysis of The OpenFlow SDN Protocol. In: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 391–396. IEEE (2017)
26. Bianco, A., Birke, R., Giraudo, L., Palacin, M.: Openflow Switching: Data Plane Performance. In: 2010 IEEE International Conference on Communications, pp. 1–5. IEEE (2010)
27. Costa, L.C., Vieira, A.B., e Silva, E.D., Macedo, D.F., Vieira, L.F., Vieira, M.A., Junior, M.D., Batista, G.F., Polizer, A.H., Goncalves, A.V., Gomes, G.: OpenFlow data planes performance evaluation. Perform. Eval. **147**, 1021 (2021)
28. Kuźniar, M., Perešíni, P., Kostić, D., Canini, M.: Methodology, measurement and analysis of flow table update characteristics in hardware openflow switches. Comput. Netw. **136**, 22–36 (2018)
29. Rygielski, P., Seliuchenko, M., Kounev, S., Klymash, M.: Performance Analysis of SDN Switches With Hardware and Software Flow Tables. In: VALUETOOLS (2016)
30. Siddiqui, A.J., Boukerche, A.: On The Impact of DDoS Attacks on Software-defined Internet-of-vehicles Control Plane. In: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 1284–1289. IEEE (2018)
31. Iperf. https://iperf.fr/. Accessed 25 March 2023
32. Fontes, R.R., Afzal, S., Brito, S.H., Santos, M.A., Rothenberg, C.E.: Mininet-WiFi: Emulating Software-defined Wireless Networks. In: 2015 11th International Conference on Network and Service Management (CNSM), pp. 384–389. IEEE (2015)
33. Sangodoyin, A., Sigwele, T., Pillai, P., Hu, Y.F., Awan, I., Disso, J.: DoS Attack Impact Assessment on Software Defined Networks. In: International Conference on Wireless and Satellite Systems, pp. 11–22. Springer (2017)
34. Abdullah, A.F., Salem, F.M., Tammam, A., Azeem, M.H.A.: Performance analysis and evaluation of software defined networking controllers against denial of service attacks. J. Phys.: Conf. Ser. **1447**, 012007 (2020)
35. Dayal, N., Srivastava, S.: Analyzing Behavior of DDoS Attacks to Identify DDoS Detection Features in SDN. In: 2017 9th International Conference on Communication Systems and Networks (COMSNETS), pp. 274–281. IEEE (2017)
36. Alharbi, T., Layeghy, S., Portmann, M.: Experimental Evaluation of The Impact of DoS Attacks in SDN. In: 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1–6. IEEE (2017)
37. Mladenov, B.: Studying the DDoS Attack Effect over SDN Controller Southbound Channel. In: 2019 X National Conference with International Participation (ELECTRONICA), pp. 1–4. IEEE (2019)
38. Kandoi, R., Antikainen, M.: Denial-of-service Attacks in OpenFlow SDN Networks. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1322–1326. IEEE (2015)

39. Pascoal, T.A., Fonseca, I.E., Nigam, V.: Slow denial-of-service attacks on software defined networks. Comput. Netw. **173**, 107223 (2020)
40. Singh, M.P., Bhandari, A.: New-flow based DDoS attacks in SDN: taxonomy, rationales, and research challenges. Comput. Commun. **154**, 509–527 (2020)
41. Singh, J., Behal, S.: Detection and mitigation of DDoS attacks in SDN: a comprehensive review, research challenges and future directions. Comput. Sci. Rev. **37**, 100279 (2020)
42. Lin, C., Wu, C., Huang, M., Wen, Z., Zheng, Q.: Performance evaluation for SDN deployment: an approach based on stochastic network calculus. China Commun. **13**(Supplement 1), 98–106 (2016)
43. Ambrosin, M., Conti, M., De Gaspari, F., Devarajan, N.: Amplified Distributed Denial of Service Attack in Software Defined Networking. In: 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–4. IEEE (2016)
44. Sun, X.S., Agarwal, A., Ng, T.E.: Controlling race conditions in openflow to accelerate application verification and packet forwarding. IEEE Trans. Netw. Serv. Manage. **12**(2), 263–277 (2015)
45. TCPReplay. https://tcpreplay.appneta.com/. Accessed 25 March 2023
46. Bonesi. https://github.com/Markus-Go/bonesi. Accessed 25 March 2023
47. Candela. https://www.candelatech.com/. Accessed 25 March 2023
48. CAIDA traffic traces. https://www.caida.org/catalog/datasets/ddos-20070804_dataset/. Accessed 25 March 2023
49. DDoS Evaluation Dataset (CIC-DDoS) (2019). https://www.unb.ca/cic/datasets/ddos-2019.html. Accessed 25 March 2023
50. Rohith, R., Moharir, M., Shobha, G., : SCAPY-A powerful interactive packet manipulation program. In: 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), pp. 1–5. IEEE (2018)
51. NOX. https://github.com/noxrepo/nox-classic. Accessed 25 March 2023
52. Open vSwitch. https://www.openvswitch.org/. Accessed 25 March 2023
53. OpenStack. https://www.openstack.org/. Accessed 25 March 2023
54. Aruba 2920. https://www.arubanetworks.com/products/switches/access/. Accessed 25 March 2023
55. Ching-Hao, C., Lin, Y.-D.: OpenFlow Version Roadmap. Technical report, tech. rep. (2015). http://speed.cis.nctu.edu.twydlin/miscpub
56. Jawaharan, R., Mohan, P.M., Das, T., Gurusamy, M.: Empirical Evaluation of SDN Controllers Using Mininet/Wireshark and Comparison with Cbench. In: 2018 27th International Conference on Computer Communication and Networks (icccn), pp. 1–2. IEEE (2018)
57. Transmission Control Protocol. RFC Editor (1981). https://doi.org/10.17487/RFC0793. https://www.rfc-editor.org/info/rfc793. Accessed 25 March 2023
58. ONF: OpenFlow Switch Specification v1.3.0. https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf. Accessed 25 March 2023
59. Semke, J., Mahdavi, J., Mathis, M.: Automatic TCP buffer tuning. In: Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 315–323. (1998)
60. Nainar, N.K., Ramdoss, Y., Orzach, Y.: Network Analysis Using Wireshark 2 Cookbook: Practical Recipes to Analyze and Secure Your Network Using Wireshark 2. Packt Publishing, Birmingham (2018)
61. Bauer, R.: Flow delegation: Flow table capacity bottleneck mitigation for software-defined networks. PhD thesis, Karlsruher Institut für Technologie (KIT) (2020). https://doi.org/10.5445/IR/1000122318. Accessed 25 March 2023.
62. Aruba: Limiting the usage of hardware resources. https://techhub.hpe.com/eginfolib/Aruba/16.10/5200-6771/index.html#s_Limiting_the_usage_of_hardware_resources.html. Accessed 25 March 2023
63. Intel: Open vSwitch* Enables SDN and NFV Transformation. https://networkbuilders.intel.com/docs/open-vswitch-enables-sdn-and-nfv-transformation-paper.pdf. Accessed 25 March 2023

Nguyen Huu Thanh received B.S and M.Sc in Electrical Engineering from Hanoi University of Science and Technology, Vietnam in 1993 and 1995, respectively. In 2002, he received his PhD with summa cum laude in Computer Science from the University of Federal Armed Forces Munich (Germany). From 2002 to 2004 he has been with the Fraunhofer Institute for Open Communication Systems (FOKUS) in Berlin, Germany. From 2004 until now he has been working as associate professor in the School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Vietnam. His research interests include QoS/QoE, network security, the Future Internet, energy-efficient networking and edge-cloud computing. He is the leader and member of several national and international research projects on network security and the Future Internet. He was the Chair of IEEE Vietnam Section (2013–2018) and currently the Vice-President of the Radio Electronics Association of Vietnam (REV).

Nguyen Ngoc Tuan received BEng degree in Systems of Information and Communication of Program of Excellent Engineers in Vietnam (PFIEV) and MSc degree in Telecommunications Engineering from Hanoi University of Science and Technology in 2009 and 2016, respectively. He is currently a PhD student at Hanoi University of Science and Technology, Vietnam. His research focuses on wireless sensor network, adhoc network, network security, new service platforms for future networks, Software-Defined Networking and the Future Internet.

Dang Anh Khoa received his B.Sc in Electronics and Telecommunications Engineering from Hanoi University of Science and Technology (Vietnam) in 2021. He received the Eiffel excellence scholarship to pursue M.Sc in Computer Networks and IoT Systems from Conservatoire National des Arts et Métiers (France) in 2022. His research interests include NFV, SDN, edge-cloud computing, and 5G mobile networks.

Le Cong Tuan received the Engineer degree in Electronics and Telecommunications Engineering from Hanoi University of Science and Technology (Vietnam) in 2022 with distinction Engineer's degree. From September, 2022 until now he has been working for Viettel High Technology Industries Corporation, Viettel Group.

Nguyen Trung Kien received his B.S and M.Sc in Electrical and Electronics Engineering from Hanoi University of Science and Technology, Vietnam, in 2019 and 2021, respectively. In 2022, he started his PhD in Computer Science at the Julius Maximilian University of Würzburg (Germany) under the scholarship of Deutscher Akademischer Austauschdienst (DAAD). His research interests include resource management in the Future Internet (NFV, SDN, edge-cloud computing) and energy-efficient networking.

Nguyen Xuan Dung received Engineering Degree in "Communications, Radio and Television" and PhD from Moscow Technical University of Communication and Informatics, Russia in 1997 and 2001, respectively. From 2002 until now Nguyen Xuan Dung has been working as Lecture in the School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Vietnam. His research interests include QoS/QoE, network planning and management, Future Internet, network security and mobile networks.

Ngo Quynh Thu is a Professor in Computer Engineering at the School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam. She received her Ph.D. in computer science in 2003 from Technical University Berlin, Germany. Her current research interests include communication protocols, wireless sensors networks, applications of the Internet of Things in different domains: agriculture, aquaculture, healthcare. She has published over 50 research articles in international journals and conference proceedings, and co-authored several books/monographs. She was a recipient of HUST Award in between 1990–1995, and the NUS Graduate School (NGS) Excellent Mentor Award in 2011.

Florian Wamser is researcher and lecturer for communication networks and cyber security at the Lucerne University of Applied Sciences and Arts in Rotkreuz, Switzerland with an affinity for monitoring, data science and data engineering. He studied at the University of Würzburg and at the Helsinki University of Technology, Finland. He received his diploma with minor subject Physics in Computer Science in 2009 (Diplom- Informatiker Univ.). During diploma thesis he worked on the topics characterization and modeling of application Internet traffic in broadband wireless access networks. In 2015 he received his PhD

(degree: Dr. rer. nat, grading: summa cum laude, award: Prize of computer science department in 2015). The title of his dissertation is "Performance Assessment of Resource Management Strategies for Cellular and Wireless Mesh Networks". In October 2022 he moved to the Lucerne University of Applied Sciences and Arts. His current research is focused on the analytical and simulative performance evaluation and optimization of cloud and communication networks and related fields.

Authors and Affiliations

Nguyen Huu Thanh[1] · Nguyen Ngoc Tuan[1] · Dang Anh Khoa[1] · Le Cong Tuan[1] · Nguyen Trung Kien[2] · Nguyen Xuan Dung[1] · Ngo Quynh Thu[1] · Florian Wamser[3]

Nguyen Ngoc Tuan
tuan.ncs16077@sis.hust.edu.vn

Dang Anh Khoa
khoa.da212454m@sis.hust.edu.vn

Le Cong Tuan
tuan.lc172901@sis.hust.edu.vn

Nguyen Trung Kien
kien.nguyen@uni-wuerzburg.de

Nguyen Xuan Dung
dung.nguyenxuan@hust.edu.vn

Ngo Quynh Thu
thunq@soict.hust.edu.vn

Florian Wamser
florian.wamser@hslu.ch

[1] Hanoi University of Science And Technology, 1 Dai Co Viet, Hanoi 11657, Vietnam

[2] University of Würzburg, Sanderring 2, Würzburg 97070, Germany

[3] Lucerne University of Applied Sciences and Arts, Suurstoffi 1, Rotkreuz 6343, Switzerland