ReCSDN: Resilient Controller for Software Defined Networks

Soomaiya Hamid, Narmeen Zakaria Bawany, Jawwad Ahmed Shamsi
Systems Research Laboratory, Department of Computer Science
FAST National University of Computer and Emerging Sciences
Karachi, Pakistan

***Abstract*—Software Defined Networking (SDN) is an emerging network paradigm that provides central control over the network. Although, this simplifies the network management and makes efficient use of network resources, it introduces new threats to network reliability and scalability. In fact, a single centralized controller is a single point of failure. Moreover, a single controller may become a performance bottleneck as processing overhead increases. Distributed SDN controller platforms improve the reliability and scalability to some extent, however they remain vulnerable to Distributed Denial of Service (DDoS) attacks, specifically on control plane. We believe that there is a need for a distributed controller framework that is capable of providing service continuity without performance degradation in case of excessive network traffic or DDoS attacks on controller. In this paper, we aim to address the vulnerabilities of SDN control plane. We propose and implement an efficient and Resilient Controller for Software Defined Network (ReCSDN). This framework is capable of detecting and mitigating DDoS attacks timely and ensures the continuity of services without performance degradation. We created an experimental test bed using Mininet to conduct extensive experiments. We deployed ReCSDN on top of Open Network Operating System (ONOS) cluster to confirm the viability of our approach. The experiment results show that with ReCSDN, control plane is not only able to withstand excessive network load but will also continue to provide services in case of any controller failure.**

Keywords—Software Defined Networking (SDN); SDN Controller security; Distributed Denial of Service (DDoS) attack; load balancing; SDN controller cluster; Open Network Operating System (ONOS)

I. Introduction

Software Defined Networking (SDN) paradigm has revolutionized the traditional networking by separating the control plane and data plane of the network. With this separation of the control plane and data plane, control logic is implemented in logically centralized controller and network switches becomes simple forwarding devices [1]. This decoupling provides several benefits which includes easier network management, increased visibility into the network, programmability, efficient use of network resources, dynamic updating of network policies [2], [3]. The centralized control plane leads to global knowledge of the network thereby providing effective resource management. Moreover, network policies can be easily configured and modified via software applications running on top of the controller. Customized network applications can be developed and deployed directly without any vendor dependency [4], [5].

Nevertheless, these core benefits that are the hype of SDN are also the main causes of concern. The centralized control plane that provides critical advantages over the traditional networking has introduced new threat vectors. First and foremost it can become the single point of failure [6]. The controller becomes the core of network and any attack, such as, DDoS attack can bring down the whole network. This vulnerability introduces new threat vector in SDN. Many approaches, such as primary backup replication mechanism and distributed controller platforms [7] exists that addresses this critical reliability issue. However, there are numerous issues with these approaches which makes it an open research problem [8], [9].

Second, the controller may turn out to be a performance bottleneck as the network size increases [8]. Whenever a new flow is initiated in the network, the OpenFlow switch forwards it to controller for deciding the suitable forwarding path. Similarly, all the unknown flows that are not recognized by the switch are sent to controller for processing. The performance of the controller is largely affected as the network grows thereby increasing the number of traffic flows. Various schemes for controller load-balancing [10] has been proposed to improve the performance of centralized controller platforms. However, due to their limited capabilities the problem remains an open research area.

Many researchers have explored the new threat vectors introduced by SDN [11], [12]. Several attacks, including DDoS attacks, and their mitigation strategies has been proposed [13]-[15] for SDN networks. However, very limited work has been done to detect and mitigate attacks specifically on SDN controllers[6]. Also, most of this work has been done for centralized controllers such as Floodlight [16]-[18] and POX [19], [20].

Keeping in view the above mentioned limitations we presume that there is a need to explore load balancing and DDoS attack vulnerabilities in distributed SDN controller platforms. We also need a framework that can detect excessive load on controllers and ensure the continuity of services without performance degradation.

To this end, we propose and describe Resilient Controller for Software Defined Networks (ReCSDN) that addresses the above mentioned problems. ReCSDN, is a novel framework that is built on top of a distributed controller environment. It provides a reliable, efficient and resilient control plane that not only overcomes the single point of failure problem but also ensures the service continuity without performance degradation. ReCSDN is able to detect the excessive network traffic coming to the controller and provides a load-balancing mechanism that ensures that performance of controller is not degraded. Excessive traffic may be generated due to DDoS attack on controller or flash crowds. In either case, the objective of ReCSDN is to provide fault tolerance and service continuity while maintaining the performance quality. ReCSDN also ensures that network latency remains consistent and does not increase as we increase the number of distributed controllers.

The main contributions of this paper are summarized below:

- We proposed and implemented ReCSDN, a reliable, efficient and resilient framework for SDN.
- We performed extensive experiments using Mininet and ONOS [21], a distributed SDN controller platform to test the effectiveness of our framework.
- We were able to detect and mitigate DDoS attack on SDN controller effectively.
- We are able to ensure quality of service performance by providing appropriate load balancing among controllers.
- We are able to provide fault tolerance by using backup controllers timely.

The rest of this paper is organized as follows. Section II comprises three sub-sections. First two sections briefly introduces Software Defined Networking and ONOS followed by a detailed review of existing research on SDN security. The proposed architecture and its implementation is discussed in Section III followed by the threat model which is discussed in Section IV. The experimental setup and results are presented in Section V and Section VI, respectively. Section VII concludes the paper.

II. Background and Related Work

We have divided this section in three sub-sections. Motivation for this research and the benefits of software defined networking over traditional networking are enlightened in the first sub-section. Next sub-section discusses ONOS, followed by the related work.

A. *Towards Software Defined Networks*

Building and administrating a computer network is an onerous task. Managing networks includes many challenges, such as, heterogeneity of network elements [22], vendor dependency [23], lack of centralized control, no programmability. Moreover configuration of complex networks which are dynamic in nature is more difficult, because of lack of automated mechanism for defining centralized policies. This creates scalability and configuration issues which makes traditional networks less innovative [24]. The network administrator have to configure each network device individually to apply network policies [25]. As the size of network increases number of devices also increases thereby increasing the administrative overhead.

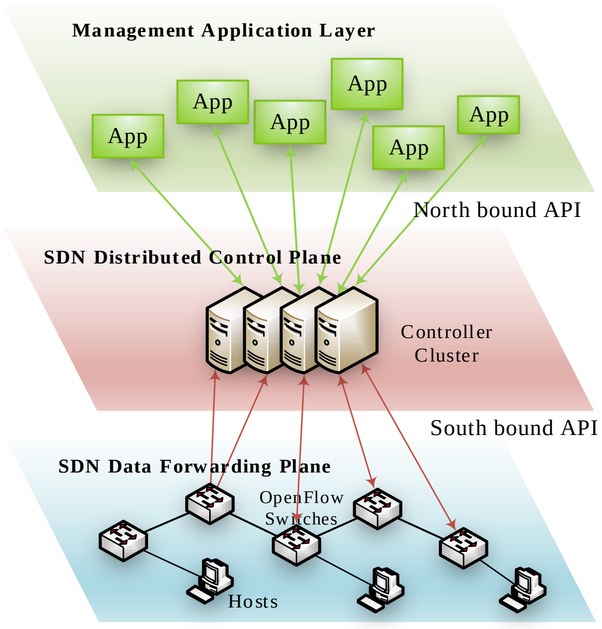

Fig. 1. Software defined network architecture.

SDN addresses the above mentioned issues by separating the control plane and the data plane as shown in Fig. 1. In SDN control plane provides a centralized control of the network. Control plane can manage the entire network centrally [12]. Major objective is to provide a centralized control over the entire network, so that all the control process and services are separated from the data forwarding tasks. Hence, the software that controls the network is decoupled from the devices that implement it [26]. Switches became simple forwarding devices that work according to the policies defined by the controller. Many open source SDN controllers has been developed which includes POX [27], NOX [28], Beacon [29] and Floodlight [30]. More recently distributed SDN controller platforms such as ONOS and OpenDaylight [31], [32] have been developed to cater the needs of large enterprise networks. We briefly discuss ONOS in the next sub-section. Apart from open source controllers, major industry leaders have also developed proprietary SDN controllers such as; HP [33], [34] and brocade [35].

Although, SDN has been gaining immense popularity since its inception, it is no silver bullet. SDN comes with its own set of vulnerabilities that were not present in traditional networks. Subsequently, after the adaptation of SDN in network infrastructures, many researchers have been questioning the security of SDN [36], [37]. The centralized control plane which has been its prominent feature has also become the major point of concern. Adversaries can launch DDoS attack on the control plane of the SDN subsequently leading to service degradation or a complete network shutdown. Similarly, performance, scalability and reliability of SDN have not been thoroughly investigated yet.

B. *Open Network Operating System*

The ONOS (Open Network Operating System) is an open source project hosted by The Linux Foundation. The software is written in Java and provides support for *distributed SDN* applications atop Apache Karaf OSGi container as shown in Fig. 2. The first version of ONOS was released in 2014. The ONOS is a distributed platform for SDN networks that caters the need of enterprise networks. The key features of ONOS includes scalability, high performance and high availability. ONOS is basically designed to operate as a cluster of nodes such that it can withstand the failure of individual nodes. ONOS overcomes the limitations of centralized SDN controllers like POX, NOX and Floodlight. It provides a high-level abstraction to application programmers by providing a platform for developers to write novel applications that can run on top of ONOS. Its model can be extended by programming variety of applications.

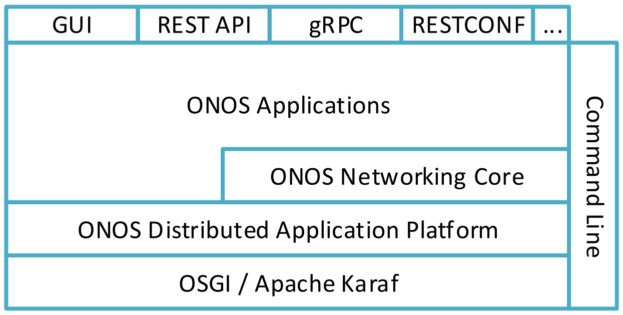

Fig. 2. ONOS software stack.

The ONOS has been used today in variety of applications ranging from multilayer network control to datacenters [38]. Major use cases of ONOS includes CORD (Central Office Re-architected as a Datacenter [39], [40], Multi-Layer Network Control, Migrating MPLS Network, and Global Research Network Development. ONOS also provides its partner driven use cases such as Huawei Agile L3 VP, Huawei Enterprise CPE, DirectTV Multicast and NEC Transport SDN.

To ensure strong consistency ONOS adopted Atomix framework after its v1.4 release. Atomix uses RAFT consensus algorithm [41] to ensure consistency among cluster nodes. Atomix deals with distributed computing problems. In contrast with the Hazelcast [42], Atomix chooses availability over consistency. Due to this Atomix ensures that data is never lost, even in the network partitioning or complete failure.

C. *Related Work*

Security of SDN has been a point of concern since its adoption [37]. Many researchers have questioned the security of SDN itself [12]. However others have proposed SDN based security solutions [43], [44]. DDoS attack detection in SDN with the entropy variation technique was presented in [6], [18] Niyaz et al. [45] proposed a deep learning multi vector DDoS system. Fonseca et al. [46] designed CPRecovery by component organization. Another technique was AVANT-GUARD [14] which is based on complete TCP handshake mechanism. Hong et al. [47] proposed a TopoGuard technique. It focused the attack over data plane communication channel. R. Braga et al. [13] classified the flows by self-organizing maps. An inference-relation context based technique was presented by Aleroud et al. [48]. They proposed technique utilizes contextual similarity with existing attack patterns to identify DoS in an OpenFlow infrastructure. Cui et al. [49] performed attack detection by neural network techniques. Botelho et al. [50] has replicated the sheared database of the whole network state to improve reliability.

Majority of the research work discussed above is based on the centralized SDN controller. Few researchers have implemented replication between master and backup controller. When master controller fails, backup controller becomes an active controller. In contrast to existing research, we have developed a resilient framework for distributed controller environment. We emulated our network using ONOS. In our approach, all controllers in a cluster are active. If there is an attack on any of the controllers, load is distributed to other controllers within a network. The controllers share the information of flows and switches consistently. Moreover in previous research works, different SDN controllers [51] were used such as, POX [27], NOX [28], Beacon [29], and Floodlight [30], but ONOS [21] controller was not explored for the attack detection. In this paper we are creating a distributed environment using ONOS controller with Mininet emulation to detect DDoS flooding attack on the controller.

III. Proposed Approach

This section presents the design of Resilient Controller for Software Defined Network – ReCSDN. The ReCSDN is a proficient solution that efficiently detects and mitigates DDoS attack on the control plane. It is capable of providing fault tolerant and consistent services to the network without performance degradation. ReCSDN detects excessive traffic network coming to the controller and uses load balancing mechanism that ensures the reliability and performance of the control plane. The ReCSDN module runs on top of distributed controller platform. It monitors the processing load of the controller and ensures that the load is distributed to other controllers in the cluster before any controller reaches its full capacity.

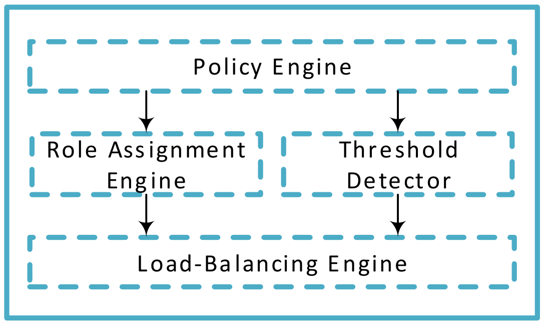

Fig. 3. ReCSDN application workflow.

ReCSDN consists of four modules as depicted in Fig. 3. The Policy Engine is used to configure the number of active and backup controllers within a cluster. Also, threshold for each controller is setup using the Policy Engine. The threshold value indicates the tolerance level of controller after which the performance of controller may be degraded. Therefore, the threshold Detector module monitors the state of controller to ensure that load of the active controller is distributed by the Load Balancing Engine before crossing the threshold. The Role Assignment Engine is used to assign the master/backup status to controllers within a cluster.

IV. Threat Model

SDN Controller is the most critical element of SDN. It serves as a centralized control of the whole network. The attack on SDN controller will result in complete shutdown of network.

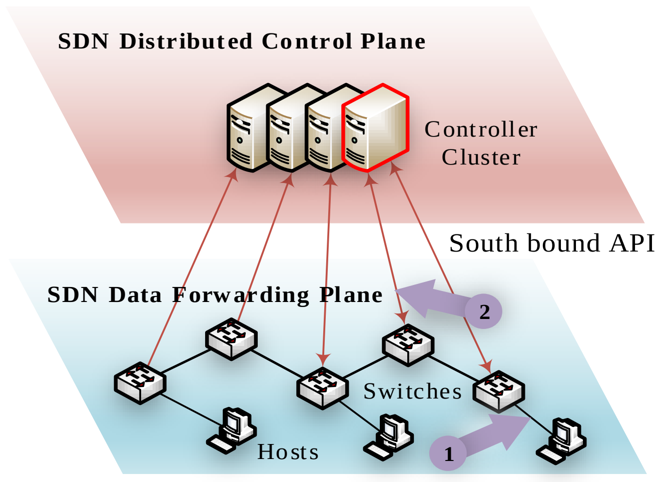

Fig. 4. Threat Model for SDN control plane.

The scope of this work is focused on DDoS attack on SDN controller. In such an attack adversaries may use compromised nodes to send unknown flows to the OpenFlow switches. These unknown flows are not recognized by the OpenFlow switches and are sent to controller for further processing. Thus, the controller is overwhelmed by the huge number of illegitimate packets and is either completely halted or results in its performance degradation.

We have considered two threat vectors that targets SDN control plane in our threat model. The two threat vector are based on generating flows that are not recognized by the switches thereby targeting the SDN controller and the communication channel between SDN control plane and data plane. Fig. 4 depicts the threat model. During a DDoS attack multiple hosts generate fake or forged traffic. Such traffic flows are not recognized by OpenFlow switches and are forwarded to controller for deciding the suitable forwarding path. This scenario not only depletes controller resources but also results in exhaustion of the communication channel between controller and the network.

V. Experimental Setup

To determine the viability of our approach, we have setup a test bed on a server with an Intel Core i7, 3.67 GHz processor and 16GB RAM running Ubuntu 14.04.5. We conducted our experiments to emulate the DDoS attack scenario on a controller using Mininet and ONOS cluster. We deployed ReCSDN module on top of ONOS cluster. We included different types of legitimate traffic to build a realistic scenario. The legitimate traffic included TCP, UDP and ICMP. The D-ITG tool [52] was used to generate the traffic and to collect performance metrics. The metrics include delay, jitter and number of packet loss.

To create DDoS attack scenario on a controller huge number of new flow requests were generated. When a new flow is received by the OpenFlow switches, it is not recognized and is forwarded to the SDN controller for deciding the transmission path. The increase in the number of new flow requests, increases the processing overhead of controller leading to performance degradation or completed denial of service. The ReCSDN module monitors the network and controllers state and ensures that load of the controller is distributed to other controllers in the network before the threshold is reached. The ReCSDN provides fault tolerance mechanism by using back controllers. These back controllers are active controllers that can also be used for load balancing in case of DDoS attack or flash crowds.

We conducted extensive experiments discussed in next section to evaluate the performance and reliability of ReCSDN.

VI. Results and Analysis

One of the key characteristics of the ReCSDN is achieving resiliency. We exploited the distributed architecture of ONOS to build a fault tolerant environment. We created a cluster of ONOS controllers that provided multiple backups for each active controller. Multiple backup controllers lead to more fault tolerance. As ReCSDN is specifically developed to work with distributed controller cluster a key aspect of characterizing the performance of ReCSDN is to analyze and compare performance at various scales. We created several scenarios to measure the response time as number of controllers in a cluster scales from 1 node to 3, 5, and 7 nodes. We observed that increasing the number of controllers within a cluster has no overhead and response time remains below 0.1ms. Fig. 5 depicts the result of experiment.

To evaluate the effect of increasing number of controllers in a ReCSDN cluster on latency we conducted multiple experiments. For each experiment we increased the number of controllers from 1 to 3, 5 and 7. We generated constant amount of TCP traffic for each experiment and noted delay and jitter. The network traffic comprises huge number of unknown flows. The ReCSDN ensured that load is distributed among the other controllers before the master controller is overwhelmed. As we increase the number of controllers in the cluster the delay decreases as shown in Fig. 6.

The latency decreased due to the consistent load distribution among the controllers. The overall performance of network improved as ReCSDN enabled load balancing before the maximum capacity of a controller is reached.

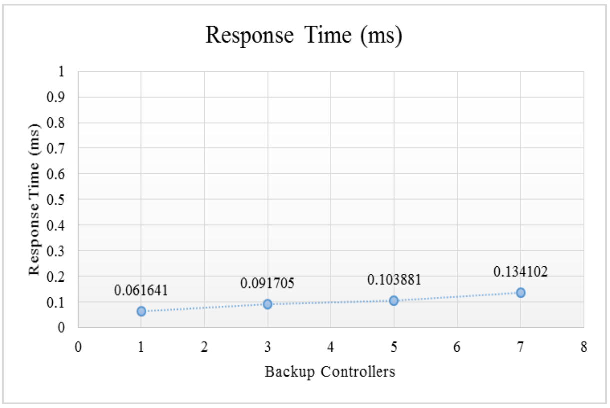

Fig. 5. Effect of adding backup controllers by calculating response time. This test was performed with varient number of backup controllers, 1, 3 ,5, and 7 respectively.

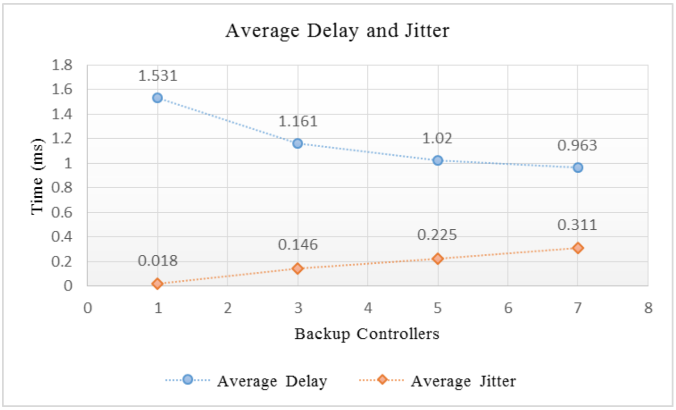

Fig. 6. Delay decreases as number of controllers increased in ReCSDN cluster.

To determine the single controller's capacity of processing maximum number of flows we performed a stress test. We flooded the controller with new flow requests, generated by pushing random intents. Intents are high-level policies that are translated by ONOS Intent Framework into installable forwarding rules. We repeatedly pushed 2000 intents till the controller halted. Fig. 7 illustrates the capacity of single controller. For our experiment, as the intent count reached 1600, the controller stopped responding. However, the capacity and performance of controller is dependent upon the configuration of physical machine on which the controller is running. After repeating the experiments number of times we choose 15000 as a threshold value for next ReCSDN experiment on this configuration. Nonetheless, the threshold value can be configured using the Policy Engine of ReCSDN whenever required.

After determining the threshold value, we launched a DDoS attack on SDN controller by pushing unknown flows in the network. We created a three controller ReCSDN cluster and started pushing intents gradually. As we moved from 1000 intents to 40,000 the ReCSDN control plane remained active without performance degradation as shown in Fig. 8. The master ReCSDN controller distributed the load to ensure the continuity of service. We also generated the legitimate traffic on the network during the attack. There were no packet losses and the response time remained consistent throughout.

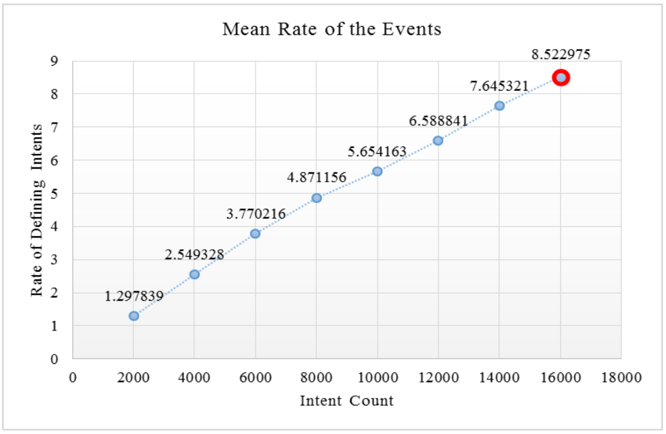

Fig. 7. Stress test for checking controller processing capability. Red indicator shows controller resources saturation point.

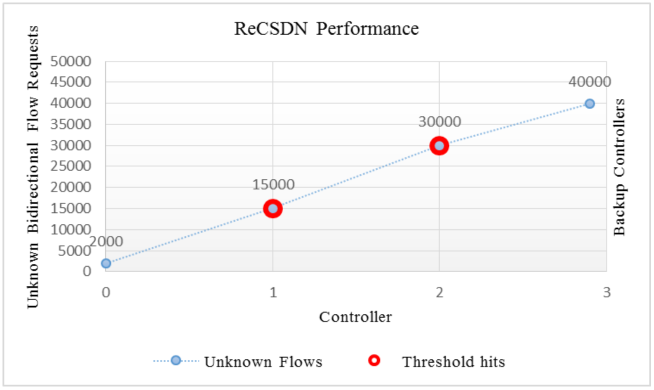

Fig. 8. ReCSDN performance evalutation.

ReCSDN is capable of provided resiliency not only in case of DDoS attack but also in case of controller failure. It improves the network performance by timely load distribution among the controllers.

VII. Conclusion and Future Work

A Software Defined Network (SDN) is an emerging network paradigm that provides central control over the network. Although the centralized control is one of the major advantages of SDN, it also brings about many critical concerns including a single point of failure in case of attacks. The central control can also become a bottleneck affecting the network's overall quality of service.

In this paper we highlighted the security threats specific to centralized control, that is, SDN control plane. We addressed the SDN's control plane issues of performance bottle neck and single point of failure.

In order to improve the performance and fault tolerance of SDN, we proposed and implemented a resilient framework-ReCSDN. Our proposed solution is not only capable of detecting excessive network traffic coming towards an SDN controller but also provides a mechanism to ensure the continuity of services in case of DDoS attack. ReCSDN uses load balancing strategy to invoke backup controllers in ReCSDN cluster to distribute and manage the load without performance degradation. We performed extensive experiments by emulating the network using Mininet and implementing ReCSDN on top of ONOS. The experiments prove that the proposed framework provides resiliency and improved performance consistency. Even though, our results are specific to the ONOS controller but the methodology we presented is general and can be applied to any distributed controller platform. In future, we intend to experiment with larger number controllers.

ACKNOWLEDGMENT

This research is supported by Higher Education Commission, Pakistan grants HEC NRPU 5946 – 2017.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” Proc. IEEE, vol. 103, no. 1, pp. 14–76, 2015.

[2] N. Bawany, J. Shamsi•, and K. Saleh, “DDoS Attack Detection and Mitigation using SDN: Methods, Practices, and Solutions,” Arab. J. Sci. Eng. Springer, Feb. 2017.

[3] B. N. Astuto, M. Mendon, X. N. Nguyen, and K. Obraczka, “A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks To cite this version :,” 2014.

[4] L. Tancevski, “SDN concept: from theory to network implementation,” in Optical Fiber Communication Conference, 2014, p. W1E.3.

[5] C. Sieber et al., “Network configuration with quality of service abstractions for SDN and legacy networks,” in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1135–1136.

[6] S. M. Mousavi and M. St-Hilaire, “Early detection of DDoS attacks against SDN controllers,” in 2015 International Conference on Computing, Networking and Communications (ICNC), 2015, pp. 77–81.

[7] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” Proc. IEEE, pp. 1–62, 2014.

[8] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in software defined networks,” Comput. Networks, vol. 71, pp. 1–30, 2014.

[9] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, “Software defined networking: State of the art and research challenges,” Comput. Networks, vol. 72, pp. 74–98, Oct. 2014.

[10] I. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “Research challenges for traffic engineering in software defined networks,” IEEE Netw., no. June, pp. 52–58, 2016.

[11] X. Wen, Y. Chen, C. Hu, and Y. Wang, “Towards a Secure Controller Platform for OpenFlow Applications,” pp. 171–172.

[12] D. Kreutz, F. M. V. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN ’13, 2013, p. 55.

[13] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in IEEE Local Computer Network Conference, 2010, pp. 408–415.

[14] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, “AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks,” in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS ’13, 2013, pp. 413–424.

[15] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, “DDoS Attack Protection in the Era of Cloud Computing and Software-Defined Networking,” 2014 IEEE 22nd Int. Conf. Netw. Protoc., pp. 624–629, Oct. 2014.

[16] J. M. Dover, “A denial of service attack against the Open Floodlight SDN controller,” vol. 21037, no. December 2013.

[17] Y. Xie and S. Z. Yu, “Monitoring the application-layer DDoS sttacks for popular websites,” IEEE/ACM Trans. Netw., vol. 17, no. 1, pp. 15–25, 2009.

[18] R. Wang, Z. Jia, and L. Ju, “An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined,” in Trustcom/BigDataSE/ISPA, IEEE, Vol. 1, 2015, pp. 310–317.

[19] T. Chin and X. Mountrouidou, “Selective Packet Inspection to Detect DoS Flooding Using Software Defined Networking (SDN),” 2015.

[20] K. Giotis, “Leveraging SDN for Efficient Anomaly Detection and Mitigation on Legacy Networks Malicious traffic growth Companies , governments and institutions are,” 2014.

[21] P. Berde et al., “ONOS: Towards an Open, Distributed SDN OS.”

[22] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, “Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges,” IEEE Commun. Surv. Tutorials, vol. 16, no. 1, pp. 369–392, 2014.

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

[24] A. Martinez et al., “Network Management Challenges and Trends in Multi-Layer and Multi-Vendor Settings for Carrier-Grade Networks,” IEEE Commun. Surv. Tutorials, vol. 16, no. 4, pp. 2207–2230, 2014.

[25] T. Benson and A. Akella, “Unraveling the Complexity of Network Management,” in e 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI’09, Berkeley, CA, USA, 2009, pp. 335–348.

[26] M. Casado, N. Foster, and A. Guha, “Abstractions for software-defined networks,” Commun. ACM, vol. 57, no. 10, pp. 86–95, Sep. 2014.

[27] Ligia Rodrigues Prete, A. A. Shinoda, C. M. Schweitzer, and R. L. S. de Oliveira, “Simulation in an SDN network scenario using the POX Controller,” in 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, pp. 1–6.

[28] N. Gude et al., “NOX: towards an operating system for networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, p. 105, Jul. 2008.

[29] D. Erickson and David, “The beacon openflow controller,” in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN ’13, 2013, p. 13.

[30] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, “Advanced study of SDN/OpenFlow controllers,” in Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia on - CEE-SECR ’13, 2013, pp. 1–6.

[31] J. Medved, R. Varga, A. Tkacik, and K. Gray, “OpenDaylight: Towards a Model-Driven SDN Controller architecture,” in Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, 2014, pp. 1–6.

[32] Z. K. Khattak, M. Awais, and A. Iqbal, “Performance evaluation of OpenDaylight SDN controller,” in 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 2014, pp. 671–676.

[33] J. Tourrilhes, P. Sharma, S. Banerjee, and J. Pettit, “SDN and OpenFlow Evolution: A Standards Perspective,” Computer (Long. Beach. Calif)., vol. 47, no. 11, pp. 22–29, Nov. 2014.

[34] A. Auyoung et al., “Corybantic: Towards the Modular Composition of SDN Control Programs.”

[35] “Brocade SDN Controller - Brocade,” 2015. [Online]. Available: http://www.brocade.com/en/products-services/software-networking/sdn-controllers-applications/sdn-controller.html. [Accessed: 23-Jul-2017].

[36] S. Shin and G. Gu, “Attacking software-defined networks,” in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN ’13, 2013, p. 165.

[37] S. Scott-Hayward, G. O’Callaghan, and S. Sezer, “Sdn Security: A Survey,” in 2013 IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–7.

[38] ONOS organization, “Use Cases - ONOS,” 2015. [Online]. Available: http://onosproject.org/use-cases/. [Accessed: 23-Jul-2017].

[39] A.-S. Ali and L. Peterson, “CORD: Central Office Re-architected as a Datacenter,” in OpenStack Summit, 2015.

[40] L. Peterson and A. Bavier, “CORD: CENTRAL OFFICE RE-ARCHITECTED AS A DATACENTER.”

[41] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in Proceedings of USENIX Annual Technical Conference, 2014.

[42] M. Johns, Getting Started with Hazelcast. Packt Publishing Ltd, 2015.

[43] N. Z. Bawany and J. A. Shamsi, "Application Layer DDoS Attack Defense Framework for Smart City using SDN," Comput. Sci. Comput. Eng. Soc. Media (CSCESM), 2016, pp. 1–9, 2016.

[44] S. Mousavi, "Early Detection of DDoS Attacks in Software Defined Networks Controller," 2014, 2014.

[45] Q. Niyaz, W. Sun, and A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," Nov. 2016.

[46] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in 2012 IEEE Network Operations and Management Symposium, 2012, pp. 933–939.

[47] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures."

[48] A. Aleroud and I. Alsmadi, "Identifying DoS attacks on software defined networks: A relation context approach," in NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 853–857.

[49] Y. Cui et al., "SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks," J. Netw. Comput. Appl., vol. 68, pp. 65–79, 2016.

[50] F. Botelho, A. Bessani, F. M. V. Ramos, and P. Ferreira, "On the Design of Practical Fault-Tolerant SDN Controllers," in 2014 Third European Workshop on Software Defined Networks, 2014, pp. 73–78.

[51] M. P. Fernandez, "Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive," in 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), 2013, pp. 1009–1016.

[52] D. Manual, A. Botta, W. De Donato, A. Dainotti, S. Avallone, and A. Pescap, "D-ITG 2.8.1 Manual," pp. 1–35, 2013.