Performance Evaluation of Software Defined Network Controllers

Edna Dias Canedo[1][a], Fábio Lúcio Lopes de Mendonça[2], Georges Daniel Amvame Nze[2], Bruno J. G. Praciano[2,3][b], Gabriel P. M. Pinheiro[3] and Rafael T. de Sousa Jr.[2][c]

[1]*Department of Computer Science, University of Brasília (UnB), Brasília-DF, Brazil*

[2]*Cybersecurity INCT Unit 6, Decision Technologies Laboratory — LATITUDE, Electrical Engineering Department (ENE), Technology College, University of Brasília (UnB), Brasília-DF, Brazil*

[3]*Department of Mechanical Engineering, University of Brasília (UnB), Brasília-DF, Brazil*

Keywords: Software Defined Networks, Virtualization, Network Topology, Software Defined Networks Controller.

Abstract: The increasing digitization of various industrial sectors, such as automotive, transportation, urban mobility, telecommunications, among others, reveals the need for using point-to-multipoint communications services, such as constantly updating software and reliably delivering messages users, as well as optimizing the use of hardware and software resources. The implementation of Software Defined Networks (SDN) provides the network with the flexibility and programming capabilities needed to accurately and reliably support point-to-multipoint distribution services. This paper presents an account of the activities implemented to produce the performance evaluation of the SDN controllers. We identified the metrics used in the literature to perform performance evaluation of SDN controllers. The mechanisms adopted were used to describe the performance evaluation environment and the respective operating processes of the controllers. Besides, a methodology is proposed to perform the performance evaluation of the controllers.

1 INTRODUCTION

Some serious problems arise in the current networks, such as the limitations of standardized equipment that runs proprietary software, the high demand of routing tables, the complexity of various protocols, the large number of applications executed by users which create network bottlenecks. To mitigate these problems, a new concept of network architecture, called Software Defined Network (SDN) (Chen, 2019), (Karimzadeh et al., 2014) has emerged.

SDN is increasingly gaining ground in the context of digital transformation and new technology capabilities, because its resource provisioning is not much affected by hardware constraints as SDN control system is centralized allowing intelligent monitoring to adapt the network automatically according to traffic conditions (Hohlfeld et al., 2018). In short, it provides a way to scan the network (Ochoa-Aday et al., 2019) and to consequently adapt the network services.

Therefore, the SDN design is an architecture of networking between computers, whose main difference to traditional models is that SDN allows centralized control of the network through software applications. In practice, this helps the operator to manage the entire network more consistently, regardless of the underlying communications technology that is used (Ochoa-Aday et al., 2019).

The SDN paradigm breaks vertical integration by radically separating the packet forwarding and the control plans, providing applications with a centralized and abstract view of network distribution (Jung et al., 2019). This architecture allows software developers to build new network deployment methods tailored to different user needs to optimize the performance of their applications and consequently optimize their tasks (Chen, 2019). Data and control plans are interconnected by public interfaces and allow direct programming of the control plan. This way, all network policies can be implemented and managed logically centrally on one controller, enabling network management as a single, programmable entity (Ai et al., 2019).

This paradigm has been gaining ground and great attention from researchers and major computer network industries (Alharbi and Portmann, 2019), (Cerroni et al., 2019), (Umair et al., 2019), (Rasool et al.,

[a] https://orcid.org/0000-0002-2159-339X
[b] https://orcid.org/0000-0002-7423-6695
[c] https://orcid.org/0000-0003-1101-3029

Canedo, E., Lopes de Mendonça, F., Nze, G., Praciano, B., Pinheiro, G. and Sousa Jr., R.
Performance Evaluation of Software Defined Network Controllers.
DOI: 10.5220/0009414303630370
In *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020)*, pages 363-370
ISBN: 978-989-758-424-4

2019), (Din et al., 2019).

The widely used and consolidated standard for SDN is OpenFlow (Poncea et al., 2019). In SDN, all intelligence is logically centered on software-based controllers (control plane) and network devices become simple packet forwarding elements (data plan) that can be programmed through an open interface provided by OpenFlow (Poncea et al., 2019).

This paper presents a proposal for a methodology for performing the performance evaluation of SDN controllers. The proposed methodology will be implemented and used by the projects developed in the Latitude research laboratory and will allow the performance monitoring of the resources used.

The remainder of this paper is organized as follows. Section 2 presents the background. Section 3 describes the proposed model for the comparison and performance of SDN controllers. Section 5 concludes the paper.

2 BACKGROUND

SDN arose due to the need to automate, scale and optimize networks, in order to better deal with applications coming from the public cloud, from private storage services and also from databases, that is, it was developed to follow the changes by which service providers and operators have faced in recent years (Alonso et al., 2019).

The SDN keep traffic flowing and assist with near-instant problem resolution, offering cost savings as costs are similar to consumption, generating long-term savings (Galán-Jiménez et al., 2018).

SDN also help improve the quality of services and/or products offered by organizations to end consumers by enabling centralized cloud monitoring, configuration, management, service delivery, control, and automation (Schwabe, 2018).

SDN controller performs network-intensive operations, and performance is a factor that deserves attention in the network role virtualization scenario, especially considering the virtualization overheads that are already widely investigated.

In the field of SDN controllers, the two main performance metrics widely considered in various investigations are throughput and latency(Lai et al., 2019; Li et al., 2019). Throughput refers to the performance of the network function, namely, the amount of OpenFlow streams the controller is capable of processing per second, while latency measures the controller response time to a PACKET _IN message received from a switch (Lai et al., 2019)(Li et al., 2019).

Some other secondary metrics may also be considered, such as:

- CPU Performance – A set of network functions are sensitive to throughput and latency because they essentially operate with packet send/receive. However, computing is also a critical factor for the performance of much of the virtualization of network functions, so it is necessary to analyze the percentage of system CPU utilization, particularly in virtual environments that can impose considerable processing degradation and limit the overall performance of the network function.
- Memory Performance – Managing RAM utilization efficiently is a critical factor in the performance of any computer system, especially in scenarios where resources are limited. Memory allocation aspects need to be considered in performance evaluation of virtualization of network functions because overall system performance can be negatively affected if poorly managed RAM results in secondary (slower) memory accesses. This situation gets even worse in virtualized scenarios because disk I/O is still a big performance bottleneck for virtualized systems.

Software Defined Networks is a paradigm that breaks vertical integration with the radical separation of packet forwarding and control plans, providing applications with an abstract centralized view of the distributed state of the network (Kantor et al., 2019).

Data and control plans are interconnected by public interfaces and allow direct programming of the control plan. This way, all network policies can be implemented and managed logically centrally as a whole on the controller, enabling network management as a single, programmable entity. Figure 1 presents the architecture of a SDN (Bozakov, 2016).

The controller is the main element of software-defined networks because it centralizes all control and management of the network. Thus, performance is a critical factor in SDN and has been the subject of much research (Bholebawa and Dalal, 2018).

In production networks, the controller manages an OpenFlow network of real switches. However, in experimental scenarios, when implementation of a real structure is not possible, tools that emulate switches are generally used, such as the Cbench tool, much explored in OpenFlow controller performance investigation (Bholebawa and Dalal, 2018).

2.1 Related Works

Since the development of SDN, many comparisons between SDN controllers have been made, mainly ad-

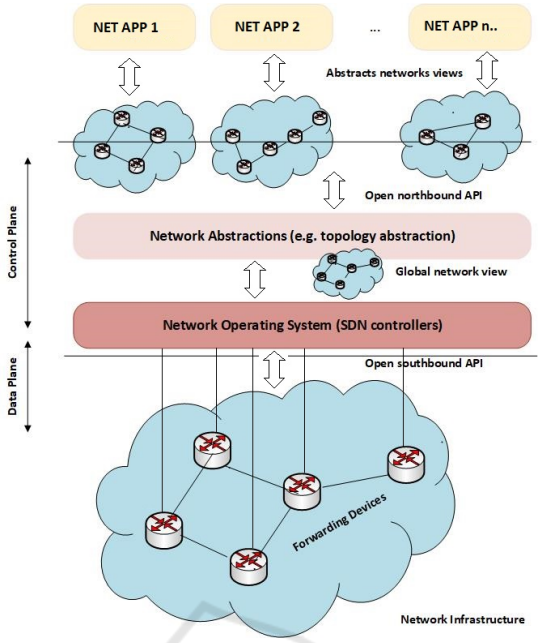

Figure 1: Architecture SDN (Bozakov, 2016).

dressing a comparative analysis of Throughput and latency using the Cbench (Laissaoui et al., 2015) (Khattak et al., 2014).

Works typically compare various open-source controllers, providing a quantitative analysis of throughput and latency. Scenarios often include varying the number of switches and threads to assess controller scalability (Khattak et al., 2014).

Tootoonchian et al. (Tootoonchian et al., 2012) have pioneered comparative studies of SDN controllers, the authors conducted a comparative analysis of the effectiveness of open source controllers.

Zhao et al. (Zhao et al., 2015) conducted a comprehensive performance evaluation of open source SDN controllers, also assessing how system configurations can interfere with controller performance.

Salman et al. (Salman et al., 2016) performed an achievement appraisal of the latest controllers such as ONOS and Libfluid-based (raw, base) controllers using Cbench (Jawaharan et al., 2018) as a testing tool and concluded that the choice of the ideal driver depends on the user's criteria. Some work goes beyond comparing controller performance to focus on benchmark tools.

Erickson (Erickson, 2013) addresses the performance evaluation of controllers and their relationship to the programming language used to develop them.

Alencar et al. (Alencar et al., 2014) inscribe a comparative study between two Java-based controllers (Floodlight and Beacon) from a different perspective: software aging. In this study, controllers are compared primarily from a memory leak standpoint, with Beacon outperforming Floodlight with less memory consumption.

Rashma and Poornima (Rashma and Poornima, 2019) proposed a rapid SDN prototyping of Single Controller (SC) and Multi Controller (MC) architecture on mininet, a programming flexible simulator. The emulation displays a Multi Controller (MC) SDN architecture which improves efficiency and exhibits competence in handling scalable network on contrary to Single-Controller (SC) architecture.

The proposed work demonstrates establishing user-defined controllers on inbuilt Open Flow controllers, which also brings intra-cluster and inter-cluster communication in a hierarchical network. They empirically ratify the scalability of the network by increasing the number of host nodes (Rashma and Poornima, 2019).

de Jesus et al. (de Jesus et al., 2014) analyze recent security proposals derived from the use of SDN and verify if its usage can help the increase of confidence, security and privacy in cloud computing environments. Furthermore, the authors approach concerns regarding security introduced by the SDN architecture and how they may compromise cloud services.

3 SDN CONTROLLER PERFORMANCE COMPARISON MODEL

This section presents the proposed environment for the performance evaluation experiment developed with an SDN controller implemented as a virtual network function. We use the SDN Cbench controller benchmark (Jawaharan et al., 2018), (Laissaoui et al., 2015) to gauge the performance of the network function deployed directly on the host and on KVM and XEN virtual machines to measure performance degradation caused by the virtualization layer.

Xen is an x86 virtual machine monitor based on a virtualization technique called paravirtualization, which has been introduced to avoid the drawbacks of full virtualization by presenting a virtual machine abstraction that is similar but not identical to the underlying hardware. Xen does not require changes to the application binary interface, and hence no modifications are required to guest applications (Armstrong. and Djemame., 2011). The XEN approach is very similar to that of KVM as is based considerably on the works of the XEN developers. XEN supports block devices through a hypercall that makes use of an altered version of the operating system.

The VNF metrics collected in the experiments were throughput, response time, CPU consumption and RAM. Table 1 presents a summary of the performance evaluation criteria of the SDN.

Table 1: SDN Performance Metrics (Lai et al., 2019; Li et al., 2019).

ID	Metric
01	Throughput
02	Latency
03	CPU performance
04	Memory Performance

To simulate a real SDN scenario, the experiments were performed in a peer-to-peer network topology such that a traffic generator transmits OpenFlow streams to a controller hosted on a dedicated server, as presented in Figure 2.

The client and server are interconnected by a 1 Gbps network link. Floodlight has been set to default according to your guidelines. Regarding the execution environment, we have used the Java language. The client machine has two (2) processors with 12 GB of RAM and 500 GB of the hard disk. The virtual machines have been configured to be identical to the base operating system (without virtualization) that housed the physical network function for a better balance in comparing network functions. For each experiment, several tests were performed.

Figure 2 shows the SDN controller deployment environment for performing Proposed Model Validation.

For each experiment, 30 tests were performed with duration of 500 seconds each, using 100 MAC addresses per switch emulated so that both the average performance for each second and the average of the generated files were stored in text files. This scenario is based on the experiments performed by Zhao et al (Zhao et al., 2015).

The validation of the proposed model for the experiments involves a set $C = \{1,2,3,4,5,6\}$ of scenarios, a set A of test environments that correspond to the network functions, a set M of variations of each environment that corresponds to the Cbench (factor) modes and the S set of switches. Considering the sets A = {physical function, VNF-KVM, VNF-XEN}, M = {throughput, latency} and $S = \{1,2,4,8,16,32\}$, the number of experiments performed to validate the proposed model must be equal to 36.

In environment 1: physical network function, the SDN controller must be installed directly over the operating system, simulating the same architecture as a dedicated hardware device, thus constituting a non-virtualized architecture that we may call a native scenario (Klosowski and Fiorese, 2019), as can be seen in Figure 3 (a).

In this scenario, its necessary to remove 8 GB of RAM from the server for a fair comparison to the virtual network function in the VNF-KVM and VNF-XEN scenarios, since the virtual machines have been configured with 8 GB of RAM. All processor cores were used in this scenario.

In environment 2: VNF-KVM, the network role must be virtualized on the standard KVM, which uses full hardware-assisted virtualization, as shown in Figure 3 (b). The emulator used must be QEMU 2.1.2. In carrying out the experiments in this scenario, it was necessary to replace the RAM, totaling 12 GB in the physical machine because the VNF must have the same amount of memory as the physical network function (8 GB), allowing the same configuration of functions Network.

In environment 3: VNF-XEN, the network function was virtualized on the XEN $4.4 - amd64$ hypervisor in the default configuration, namely using paravirtualization. XEN disables automatically the KVM hypervisor when configuring XEN at system boot, preventing KVM from consuming resources and causing inconsistencies in performance evaluation. Figure 3 (c) shows the VNF-XEN deployment environment.

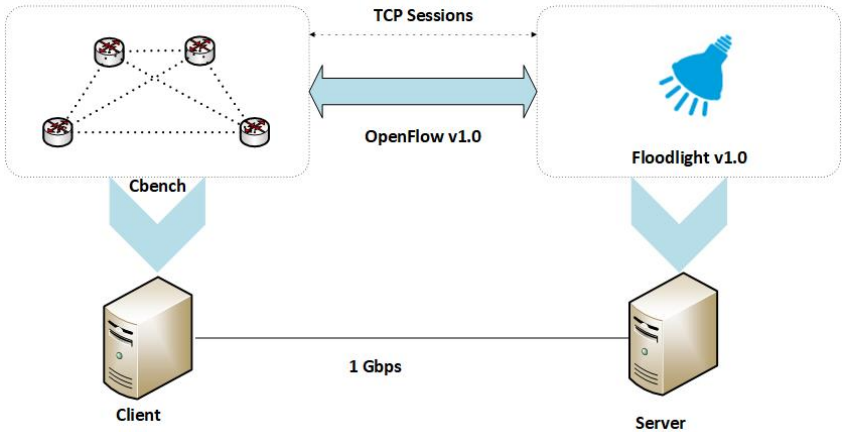

Figure 2: Controller Deployment Environment.

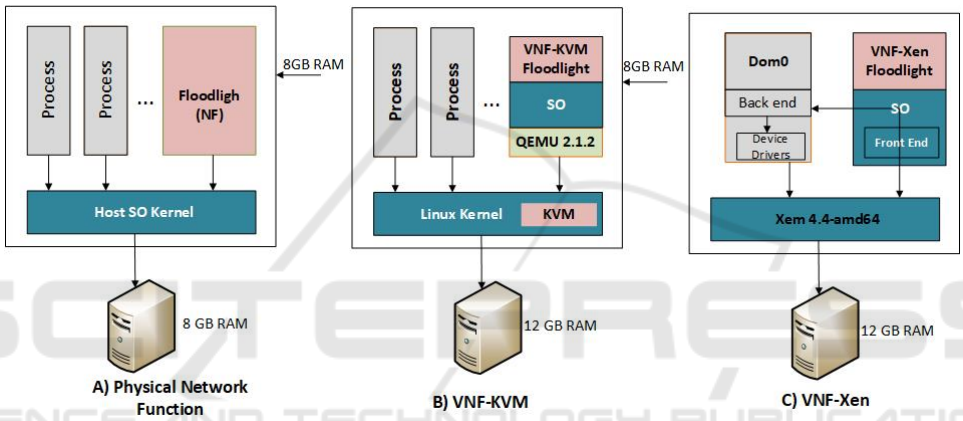

Figure 3: VNF-XEN Deployment Environment.

3.1 SDN Performance Assessment Process Mapping

During the performance evaluation of SDN, Cbench was configured to emulate a specific number of switches capable of sending streams to the controller installed on the server for a certain period. At the end of each performance validation, the data generated by Cbench was collected.

Simultaneously with the execution of Cbench, the CPU and RAM data of the physical and virtual network functions were recorded by the server. It is important to note that when initiating a benchmarking, it is important to run a script to send commands to the server, instructing it in its network function (SDN Controller) host CPU and memory monitoring activities. This script has stored all the data configured to evaluate SDN performance.

The developed script should be responsible for synchronizing the distributed system components of the SDN benchmarking process and for emptying the RAM, system buffers, and memory buffers at each start of the performance monitoring process.

The script should also have the function of writing controller performance data collected by Cbench to monitoring files on the server. Files generated during the monitoring process should store monitoring data on the network (controller) performance and performance of physical and virtual network functions related to system CPU and RAM utilization.

The proposed SDN performance evaluation process for the experiments should follow the following steps, as shown in Figure 4.

1. Define the number of switches: The performance evaluation must be planned and the switches that will participate in the experiment must be defined and configured, according to the proposed scenario and need of the evaluation project;

2. Configure hardware and software features: All features that will participate in the experiment must be configured as planned in activity 1 of the

process;

3. Define the tools to be used during monitoring, on both client and server: The tools should be defined according to plan and criteria adopted during the experiment;
4. Define performance monitoring metrics: As presented in Table 1, in the literature there are proposed metrics and they should be adopted and updated, according to the investigated scenario;
5. Define the data collection script: In benchmarking, during controller execution, a data collection script must be designed and installed on the client to trigger experiments and data collection.

   The script authorized the start of Cbench and the data monitoring tool that will be adopted in the experiment. The script must be installed on the server to send SSH commands to the server instructing it in its network function (SDN Controller) CPU and memory monitoring activities.

   The script should be responsible for synchronizing between the distributed system components of the experiment and for emptying the RAM, system buffers, and memory buffers at the start of a new experiment;
6. Develop SDNs performance reports: At the end of each experiment, reports should be generated from the results of the data collection scripts.

3.2 Performance Indicators for SDN Controllers

While network applications are generally sensitive to processing, network functions perform intensive network input/output operations. For example, a web server that receives an HTTP request can send a huge amount of CPU cycles in processing the request before sending the response to the client, while a router basically operates with a large number of forwarding decisions, using only few CPU cycles.

The SDN controller performs intensive network operations, with performance being a factor that deserves enough attention in the VNF scenario, especially if we consider the virtualization overheads that have already been extensively investigated. In the area of SDN controllers, the two main performance metrics widely considered in several scientific researches are throughput and latency. Throughput refers to the performance of the network function, that is, the number of OpenFlow flows that the controller is able to process per second. Additionally, the latency measures the response time of the controller to a PACKET_IN message received from a switch.

4 DISCUSSION AND THREATS TO VALIDITY

If any performance metric of an SDN is less than 1, it indicates that a considerable degradation of the SDN controller's performance occurred due to overloading imposed by the virtualized environment. With paravirtualization, a better result was achieved, demonstrating the high cost of emulating network devices in the host's user space used during execution.

The function of a physical network engages nearly all CPU time with the execution of the controller and the flow processing, with a reasonable number of executions, explaining the higher performance when compared to VNFs. The VNF-XEN utilized most of the processing for the controller, but the processing power engaged in the XEN environment, specially in privileged domain interactions, is considerable and results in performance degradation. When the controller is too overloaded, the physical functions as well as the virtual machines generate a great number of interruptions. Then, the processor spends much time idle while the VNF-XEN consumes more processing time with the system, but employed CPU time with flow processing is longer than the remaining functions of SDN processes.

Latency tests usually indicate a considerable degradation of performance, the the additional software layer — responsible for virtualization introduced between the hardware's host and the host's system — imply in higher network and data processing complexities. This results in higher transmission delay between the VNFs and its clients. The physical function and the virtual machines consume less processing time, as they produce more frequent interrupts per second and leave the CPU idle for a longer interval. Therefore, much of CPU time is used for virtualization and system operations.

VNF-XEN works along with the privileged domain and strives to obtain the best possible performance, including generating considerably less interruptions and shorter processor idle time. However, inter-domain operations diminish CPU performance, since XEN does not perform direct CPU scheduling for the virtual machines, once it is not a hypervisor integrated to the Kernel.

The hypervisor has the task to manage physical devices of the computer, allowing for multiple instances of virtual computers to be executed simultaneously through hardware sharing, thus optimizing resource usage. Virtual machine scheduling for CPU usage is controlled by XEN instead of by the system

Performing a swapping operations during the experiment could lead to lower controller performance

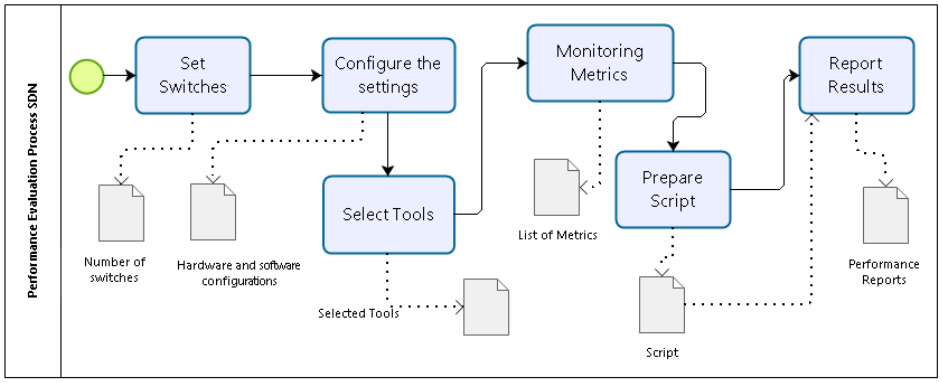

Figure 4: Performance Evaluation Process SDN.

on recording flow information and the state of accomplished connections on the hard drive, once disk I/O operations represent a critical bottleneck in performance of virtualized systems, posing a major challenge for current virtualization attempts.

As with any experiment, the configuration of the simulation environment may affect the obtained results, as it is known that, in some situations, the configurations and used resources are prone to unforeseen events. A way of mitigating this default is to run the experiment in a number of environments, with different configurations.

5 CONCLUSION

This paper presents an experiment using an SDN controller implemented as virtual functions in the KVM and XEN environments to compare the most critical performance parameters in a native scenario to verify the performance of network functions and the innovative processes linked to the performance evaluation of SDN. A simulation environment was built to test and possible SDN configurations.

The experiments showed that the virtualization of an SDN controller in the KVM and XEN environments did not overload the resources of the proposed scenario. From the proposed scenario, we defined the mapping of the performance evaluation process to be adopted in the experiments.

As future work, the proposed process will be tested in various scenarios, with experiments to collect metrics to verify if the adopted process is adequate to evaluate the performance of SDN controllers. These experiments will be important to validate and evolve the proposed process.

ACKNOWLEDGMENTS

This work was supported by the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES), Grants 23038.007604/2014-69 FORTE and 88887.144009/2017-00 PROBRAL; the Brazilian National Council for Scientific and Technological Development (CNPq), Grants 303343/2017-6, 312180/2019-5 PQ-2, BRICS2017-591 LargEWiN, and 465741/2014-2 INCT on Cybersecurity; The Brasilian Federal District Research Support Foundation (FAP-DF), Grants 0193.001366/2016 UIoT and 0193.001365/2016 SSDDC; the LATITUDE/UnB Laboratory (Grant 23106.099441/2016-43 SDN); the Ministry of Economy (TEDs DIPLA 005/2016 and ENAP 083/2016); the Institutional Security Office of the Presidency of the Republic (TED ABIN 002/2017); the Administrative Council for Economic Defense (TED CADE 08700.000047/2019-14); and the Federal Attorney General (TED AGU 697.935/2019).

REFERENCES

Ai, J., Guo, Z., Chen, H., and Cheng, G. (2019). Improving the routing security in software-defined networks. *IEEE Communications Letters*, 23(5):838–841.

Alencar, F., Santos, M. A., Santana, M., and Fernandes, S. (2014). How software aging affects SDN: A view on the controllers. In *GIIS*, pages 1–6. IEEE.

Alharbi, T. and Portmann, M. (2019). The (in)security of virtualization in software defined networks. *IEEE Access*, 7:66584–66594.

Alonso, R. S., Sittón-Candanedo, I., Rodríguez-González, S., García, Ó., and Prieto, J. (2019). A survey on software-defined networks and edge computing over iot. In *PAAMS (Workshops)*, volume 1047 of *Commu-*

nications in Computer and Information Science, pages 289–301. Springer.

Armstrong., D. and Djemame., K. (2011). Cultivating cloud computing - a performance evaluation of virtual image propagation & i/o paravirtualization. In *Proceedings of the 1st International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, pages 539–550. INSTICC, SciTePress.

Bholebawa, I. Z. and Dalal, U. D. (2018). Performance analysis of sdn/openflow controllers: POX versus floodlight. *Wireless Personal Communications*, 98(2):1679–1699.

Bozakov, Z. (2016). *Architectures for virtualization and performance evaluation in software defined networks*. PhD thesis, University of Hanover, Hannover, Germany.

Cerroni, W., Galis, A., Shiomoto, K., and Zhani, M. F. (2019). Telecom software, network virtualization, and software defined networks. *IEEE Communications Magazine*, 57(5):88.

Chen, L. (2019). *Resource Allocation in Multi-Domain Wireless Software-Defined Networks. (Allocation de ressources dans des réseaux définis par logiciels sans-fil multi-domanes)*. PhD thesis, INSA Toulouse, France.

de Jesus, W. P., da Silva, D. A., de Sousa Júnior, R. T., and da Frota, F. V. L. (2014). Analysis of SDN contributions for cloud computing security. In *UCC*, pages 922–927. IEEE Computer Society.

Din, S., Paul, A., and Rehman, A. (2019). 5g-enabled hierarchical architecture for software-defined intelligent transportation system. *Computer Networks*, 150:81–89.

Erickson, D. (2013). The beacon openflow controller. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 13–18. ACM.

Galán-Jiménez, J., Polverini, M., and Cianfrani, A. (2018). Reducing the reconfiguration cost of flow tables in energy-efficient software-defined networks. *Computer Communications*, 128:95–105.

Hohlfeld, O., Kempf, J., Reisslein, M., Schmid, S., and Shah, N. (2018). Guest editorial scalability issues and solutions for software defined networks. *IEEE Journal on Selected Areas in Communications*, 36(12):2595–2602.

Jawaharan, R., Mohan, P. M., Das, T., and Gurusamy, M. (2018). Empirical evaluation of SDN controllers using mininet/wireshark and comparison with cbench. In *ICCCN*, pages 1–2. IEEE.

Jung, O., Smith, P., Magin, J., and Reuter, L. (2019). Anomaly detection in smart grids based on software defined networks. In *Proceedings of the 8th International Conference on Smart Cities and Green ICT Systems - SMARTGREENS,*, pages 157–164. INSTICC, SciTePress.

Kantor, M., Biernacka, E., Borylo, P., Domzal, J., Jurkiewicz, P., Stypinski, M., and Wójcik, R. (2019). A survey on multi-layer IP and optical software-defined networks. *Computer Networks*, 162.

Karimzadeh, M., Valtulina, L., and Karagiannis, G. (2014). Applying sdn/openflow in virtualized LTE to support distributed mobility management (DMM). In *CLOSER*, pages 639–644. SciTePress.

Khattak, Z. K., Awais, M., and Iqbal, A. (2014). Performance evaluation of opendaylight SDN controller. In *ICPADS*, pages 671–676. IEEE Computer Society.

Klosowski, E. A. and Fiorese, A. (2019). Freecontroller: A framework for relative efficiency evaluation of software-defined networking controllers. In *ICEIS (1)*, pages 349–360. SciTePress.

Lai, Y., Ali, A., Hossain, M. S., and Lin, Y. (2019). Performance modeling and analysis of TCP and UDP flows over software defined networks. *J. Network and Computer Applications*, 130:76–88.

Laissaoui, C., Idboufker, N., Elassali, R., and Baamrani, K. E. (2015). A measurement of the response times of various openflow/sdn controllers with cbench. In *AICCSA*, pages 1–2. IEEE Computer Society.

Li, F., Xu, X., Yao, H., Wang, J., Jiang, C., and Guo, S. (2019). Multi-controller resource management for software-defined wireless networks. *IEEE Communications Letters*, 23(3):506–509.

Ochoa-Aday, L., Cervello-Pastor, C., and Fernández-Fernández, A. (2019). etdp: Enhanced topology discovery protocol for software-defined networks. *IEEE Access*, 7:23471–23487.

Poncea, O. M., Pistirica, S. A., Moldoveanu, F., and Asavei, V. (2019). Design and implementation of an openflow SDN controller in NS-3 discrete-event network simulator. *IJHPCN*, 14(1):17–29.

Rashma, B. and Poornima, G. (2019). Performance evaluation of multi controller software defined network architecture on mininet. In *International Conference on Remote Engineering and Virtual Instrumentation*, pages 442–455. Springer.

Rasool, R. U., Ashraf, U., Ahmed, K., Wang, H., Rafique, W., and Anwar, Z. (2019). Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks. *IEEE Access*, 7:34885–34899.

Salman, O., Elhajj, I. H., Kayssi, A., and Chehab, A. (2016). Sdn controllers: A comparative study. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pages 1–6. IEEE.

Schwabe, A. (2018). *Data-centre traffic optimisation using software-defined networks*. PhD thesis, University of Paderborn, Germany.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. (2012). On controller performance in software-defined networks. In *Hot-ICE*. USENIX Association.

Umair, Z., Qureshi, U. M., Cheng, T. Y., and Jia, X. (2019). An efficient wireless control plane for software defined networking in data center networks. *IEEE Access*, 7:58158–58167.

Zhao, Y., Iannone, L., and Riguidel, M. (2015). On the performance of SDN controllers: A reality check. In *NFV-SDN*, pages 79–85. IEEE.