Research Article

Benchmarking and Performance Analysis of Software Defined Networking Controllers in Normal and Failsafe Operations using Multiple Redundant Controllers

Ayman Haggag [1]

[1] Electronics Technology Department, Faculty of Technology and Education, Helwan University.
haggag@techedu.helwan.edu.eg

Article History: Received: 5 April 2021; Accepted: 14 May 2021; Published online: 22 June 2021

Abstract: Software Defined Networking (SDN) has a great impact on networks and data communication. The future of the infrastructure for networks will be different. Network administrators will no longer need to configure every router or switch, and it will be unnecessary to program or write their configurations individually. A central control device is used to control the entire network. In this research, we will build network topologies using open flow switches connected to open flow controllers that will provide flow tables for switches. We will use Benchmarking and compare the performance of different popular open flow controllers, OpenDaylight (ODL), Open Network Operating System (ONOS), RYU, Floodlight, POX, and Trema. We will show how this new infrastructure drastically improves network performance and several legacy protocols, like Spanning Tree Protocol, are rendered obsolete. In larger multiple controller networks, the network resilience against controller failure with the utilization of multiple redundant controller connections is demonstrated.

Keywords: SDN, SDN controllers, Benchmarking, Failsafe operation

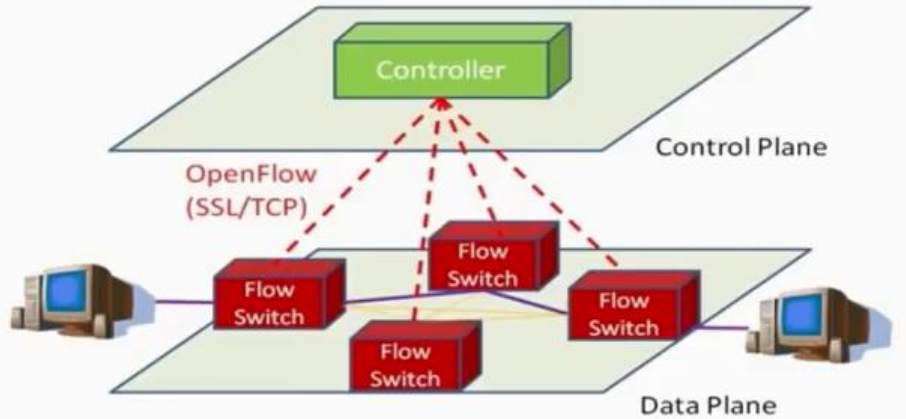
1. Introduction

The idea behind Software Defined Networking (SDN) is the separation between hardware and software similar to the concept applied to computer systems separating hardware from software (Haggag, 2019). Traditional network devices like routers and switches come with permanent proprietary preinstalled operating systems. With SDN, those devices will be unintelligent pieces of hardware, and all the intelligence and processing power will be transferred to the controller. This will provide a centralized point to manage all devices in the network. SDN will lead to new developments, such as, Software-Defined Storage (SDS) **(Gracia-Tinedo et al., 2019)**, Software-Defined Data Center (SDDS) **(Dai et al., 2017)**, Software Defined Networking Security (SDNS) **(Correa Chica et al., 2020)**, as well as the new concept of Network Functions Virtualization (NFV) **(Li et al., 2019)** that will enable the creating of virtual switches, virtual routers, virtual load balancers, virtual firewalls which would dramatically reduce out the need for those network devices and provide a great reduction in cost.

1.1. History of SDN

Martin Casado was a researcher interested in virtualization and virtual networks. In 2007, he founded a company called Nicira in California specialized in virtual networks. During his Ph.D. study at Stanford University, he worked on the development of the OpenFlow protocol as an open-source protocol. Meanwhile, Nicira company developed the first virtual switch, then called Open vSwitch (OvS) **(Gude et al., 2008)**. In 2012, VMware acquired Nicira company and Casado became the VMware CEO for Networks, Information Technology and Security. He aimed at developing an operating system that can be installed on any networking device or even a Personal Computer (PC). This could be achieved by using the OpenFlow protocol **(Sherwood et al., 2009)** that would enable a controller to manage and control hardware networking devices to forward data accordingly. SDN was then defined as the physical separation of control plane and data plane where a centralized control plane controls a set of several distributed devices as shown in Figure 1.

Figure 1. A schematic diagram of Software Defined Networking (SDN).



1.2. Real-life uses of SDN

Google used SDN since 2010 and claimed that the use of SDN greatly reduced storage expenses, routers and switches costs, and reconfiguration costs. As Google network is an expanding network, the use of SDN reduced the need for network engineers to configure new devices **(Dixon et al., 2016)**. Facebook started to endorse the SDN idea and started to manufacture switches that support OpenFlow protocol. They started with the production a switch called wedge switch, then they produced another switch with a larger number of ports, faster and higher in efficiency called 6-pack switch, those switches support the OpenFlow protocol with an operating system called F-Boss that supports OpenFlow and can be used to manage all switches centrally **(Cui et al., 2016)**. Microsoft also endorsed SDN and claimed that this reduced yearly development budget and enabled network expansion at a lower cost with full central control over network devices **(Greenberg, 2015)**. This enables us to say that SDN is already a reality starting to emerge stronger and it is expected that soon, SDN will dominate the networking industry.

1.3. Traditional Networks

In traditional networks, every networking device, such as a router or a switch, has its management plane, control plane, and data plane as shown in Figure 2.

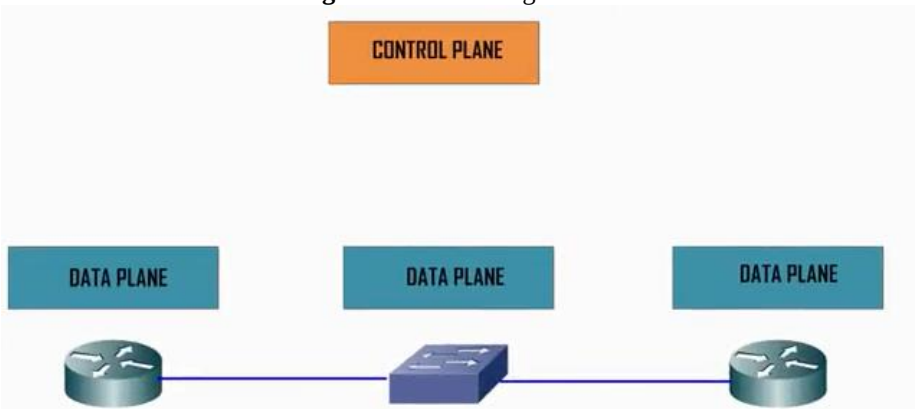
Figure 2. Every networking device has its management plane, control plane, and data plane.

For the management plane, each device needs to be programmed separately. The network administrator needs to access each device either directly using a console cable or remotely using various management protocols, such as SSH, telnet, or SNMP protocols, and every network device needs to be managed separately, one device at a time. The control plane can be considered as the brain of the device. These are the protocols that many run on routers such as routing protocols and tools used to build the MAC address table, access list, quality of service, route map or to create VLANs. The data plane consists of the physical routers and switches with their ports, as well as routing tables, created via the OSPF protocol, to enable data forwarding according to the routing table. OSPF protocol has many known stability issues **(Basu & Riecke, 2001)**.

1.4. Open SDN Concept

In SDN, devices only retain the data plane, the management plane, and the control plane are merged and removed from all networking devices and places in a central device called the Controller as shown in Figure 3.

Figure 3. SDN configuration.

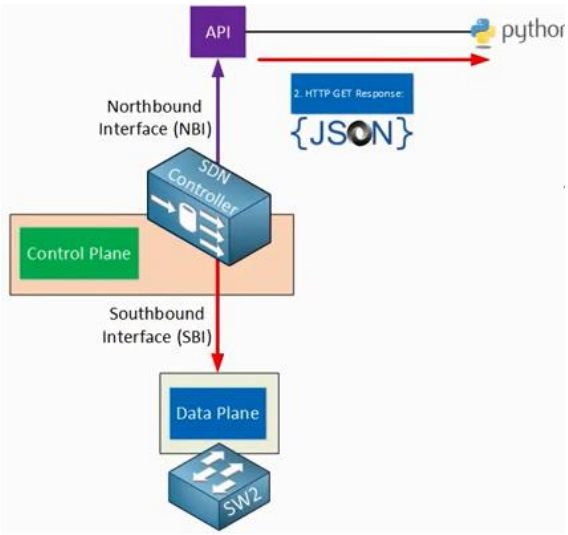


The controller is mainly a software program installed on a powerful computer that acts as a server and functions as the brain of all routers and switches in the entire network. The controller builds a flow table for the entire network instead of every router building its routing table and every switch building its own MAC address table. The controller sends this flow table to all devices in the network via the OpenFlow protocol **(Hu et al., 2014)**.

1.5. Application Programming Interface (API)

Application Programming Interface (API) is a virtual interface that defines interactions between multiple software intermediaries. It a communication language is that understood by two different parties. Applying this concept to the SDN controller, we can see that the controller utilizes what is called Northbound Interface (NBI) to help applications written in Python and JAVA interact with the controller using Extensible Markup Language (XML), or JavaScript Object Notation (JSON) **(Pham & Hoang, 2016)**. The controller also uses what is called Southbound Interface (SBI) to help the controller interact with switches using openflow, netconf, SNMP, OVSDB or opflex protocols as shown in Figure 4.

Figure 4. Northbound Interface and Southbound Interface in SDN.

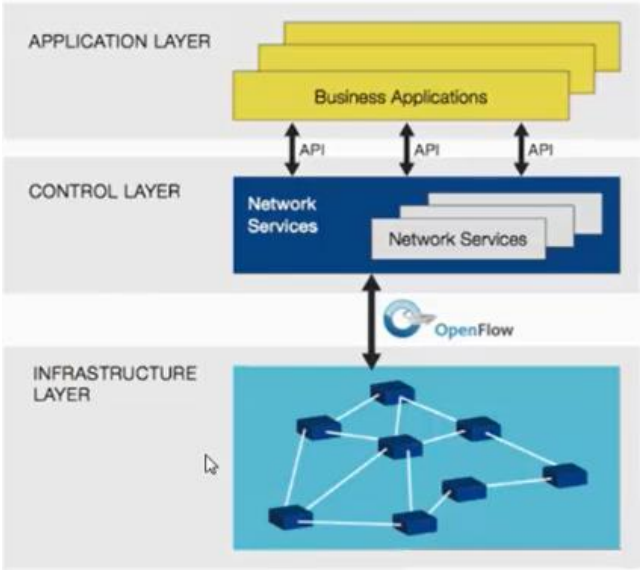


Larger networks utilize several SDN controllers, in this case, Eastbound Interface (EBI) and Westbound Interface (WBI) are used to allow controllers to communicate with each other.

1.6. 0penFlow Architecture Components

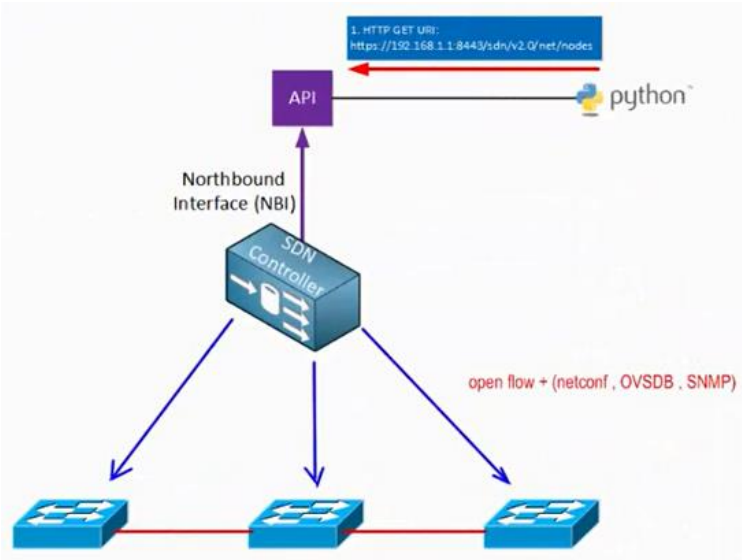
Open Flow can be viewed as an architecture or as a protocol. OpenFlow as a protocol is the protocol that enables the controller to communicate with the network hardware. OpenFlow architecture **(Jarschel et al., 2011)** consists of the infrastructure layer, control layer, and application layer as shown in Figure 5.

Figure 5. Open Flow architecture.



The infrastructure layer consists of switches that connect the network and those switches are being controlled via a Controller. This controller is the brain of the network and, at the same time, this controller is capable of being programmed with new protocols or new designs or use to deploy a new protocol to all switches using various programming languages like Python and Java as shown in Figure 6.

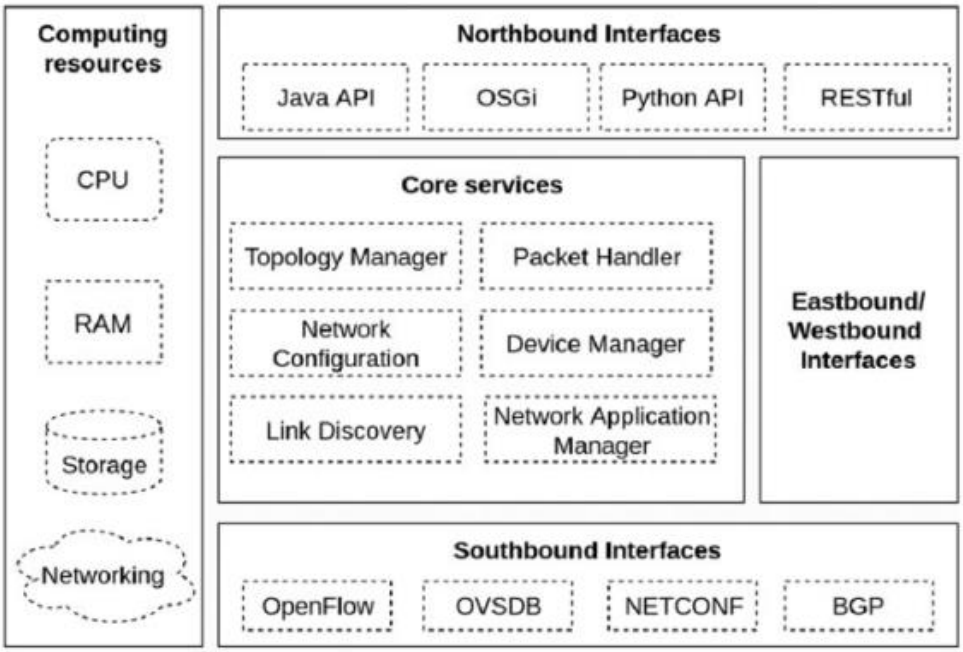
Figure 6. Open Flow Components.



1.7. The Controller

The controller must support the OpenFlow protocol. Various other protocols can be used as a communication protocol between the controller and switches, like NetConf, OVSDB, or SNMP protocols. Figure 7 shows a schematic diagram of the main components of an SDN controller **(Hoang & Pham, 2015)**.

Figure 7. Major components of the SDN controller.



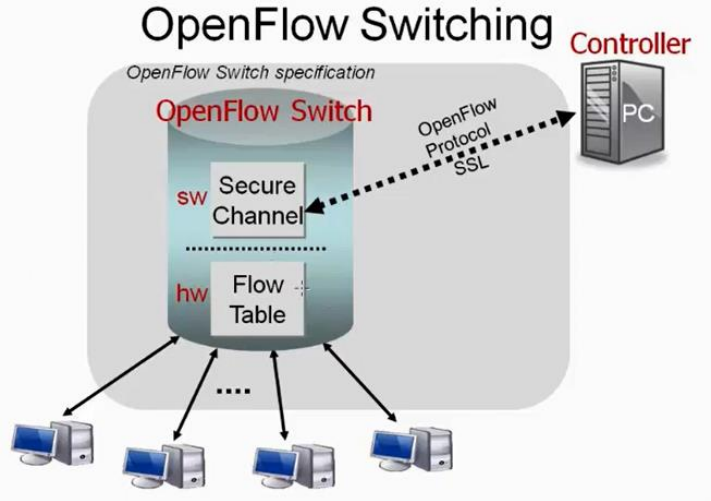
1.8. The Open Flow Switch

The Open Flow switch consists of an open flow secure channel, open flow ports, flow tables, group tables, and meter tables. Open Flow Network Foundation (ONF) is the organization responsible for the specifications and features that the Open Flow switch and OpenFlow protocol support. There two types of Open Flow switches, pure open flow switch, and hybrid switch. The pure open flow switch is a switch with no control plane and cannot make decisions and forwarding on its own. The hybrid switch is a switch with an operating system and is capable of forwarding as it has its control plane, however, it has the capability also to operate as an open flow switch that can connect to an openflow controller and receive a flow table. OpenFlow switches are now easily built using FPGA **(Liu, 2014)**.

1.9. Open Flow Secure Channel

Open flow secure channel is the communication interface between the switch and the controller. It is a TCP based session that works on port numbers 6653 or 6633 and is usually initiated by the switch. This TCP session is encrypted using Transport Layer Security (TLS) protocol or Secure Sockets Layer (SSL) protocol. The switch and the controller negotiate the open flow version, versions 1.1, 1.2. 1.3. 1.4 and 1.5 are available. The negotiation process ends with an agreement about a common version supported by both the switch and the controller to enable backward compatibility. The controller starts sending encrypted flow entries to populate the flow table in the switch. At the same time, the switch sends echo requests using the ping command to check and ensure connectivity as shown in Figure 8.

Figure 8. Open Flow Secure Channel.



1.10. Open Flow Ports

Open flow ports can be either physical or logical. A physical open flow port is a real port connecting a switch directly to the controller. A logical open flow port is when the switch is not directly connected to the controller, however, a logical open flow port is a logical connection established through other switch or switches to communicate with the controller.

2. Benchmarking SDN Controllers

Several benchmarking tools exist for evaluating SDN controllers **(Andrade et al., 2016)**. CBench **(Sherwood & Kok-Kiong, 2010)** is an open-source benchmarking tool designed for the evaluation of OpenFlow SDN controllers. Its operational model simulates SDN switches that send requests (Packet_IN) to the controller using OpenFlow protocol. CBench uses Latency and Throughput as the two evaluation metrics. HCprobe is the extension of CBench to provide additional capabilities in evaluation. WCBench is another extension of CBench built-in Python that offers additional aspects of evaluation and statistics.

PktBlaster **(World & Emulation, n.d.)** is another benchmarking tool that tests large scale SDN networks. It can measure throughput and latency for different test profiles as well as the size of the switch's buffer and flow tables.

OFNet **(Shankar, 2016)** benchmarking tool can generate different types of topologies with a traffic generator that produces different types of traffic. It can measure performance characteristics such as CPU utilization, flow entries, flow failure, Round Trip Time (RTT), and latency of flow setup. We give a summary of the benefits and drawbacks of used benchmarking tools in Table 1.

Table 1. Advantages and drawbacks of used benchmarking tools.

Name	Advantages	Drawbacks
Cbench	Open-source. Provides multiple parameters for achieving throughput and latency tests.	Only supports OpenFlow version 1.0. It has not been updated.
PktBlaster	Supports larger networks.	Only supports standard topologies.
OFNet	Contains a built-in traffic generator. Supports the building of custom topologies.	Variant results according to topology used

3. Simulation and Experiments

SDN is simulated using Linux based Mininet emulator running on a 64-bit Ubuntu 14.04 that has many SDN software and tools installed, running within VMWare workstation emulation to create and run example topologies. Mininet includes Open vSwitch 2.3.0 with support for Openflow 1.2, 1.3 and 1.4, and LINC switch. An

implementation of OpenDaylight, ONOS, RYU, Floodlight, Floodlight-OF1.3, POX, and Trema are used. Wireshark network protocol analyzer, Wireshark 1.12.1, with native support for OpenFlow parsing is used to capture and analyze traffic. CBench, PktBlaster, and OFNet benchmarking tools are used for benchmarking the performance of tested SDN controllers in term of throughput and latency for several network topologies.

3.1 Throughput Metrics

Throughput is a measurement of the rate of processing flow requests by the controller. It is measured by the number of packet_in messages sent and the corresponding packet_out messages received per unit time.

3.2 Latency Metrics

Latency is time between the packets are sent to the controller and the response is received at the vSwitch.

4. Comparative Analysis of SDN vs Traditional Networks

Traditional networks as described earlier suffer from several limitations such as data forwarding is time-consuming, error-prone, and has high conversion times, besides, this setup limits automation, innovation, and the ability to connect devices from different vendors together. All devices in traditional networks are closed systems, it is not possible to upgrade a device or try a new protocol and new devices need to be purchased that support the new protocol.

Benefits of SDN resulting from separating the control plane from the data plane are summarised as, the response to link failure or node failure if faster, loop avoidance is simpler, management is simpler, less time consuming, money-saving, support for automation, innovation, and agility.

In traditional networks, network failure notifications need to propagate from one device to another until all devices are notified of that failure, however in SDN, as soon as a failure occurs, the controller is notified, and the controller immediately notifies all other devices about that failure to update the flow table.

Loop avoidance is simpler with SDN as in the presence of the controller, the occurrence of loops can be mitigated without the need for the Spanning Tree Protocol as the controller has a complete map of the entire network.

Management is simpler with SDN as the controller is a centralized point that can be used to manage the entire network. It is possible through the controller to enable a specific protocol, for example, RIP on specific router ports, create VLANs on switches, or create an access list and update appropriate devices using OpenFlow protocol.

Less time consuming and money-saving with SDN as the controller supports Application Programming Interface (API) virtual interface that enables developers to write their codes for new protocols using JAVA or Python programming languages and deploy it on the controller, the controller in turns deploys the new protocol on routers and switches using OpenFlow, thus saves the money of purchasing new hardware and software as Open Day Light (ODL) is an open-source OpenFlow controller free for anyone to use and also extends network capabilities beyond any limitations through development of new protocols and supports innovation.

Automation is supported with SDN as the controller can be used to automatically apply network policies to all routers and switches all at once. This also gives rise to increased network agility, as it is very simple to change network policies from time to time according to dynamic network demands and conditions.

The problem of having a centralized point of control for the entire network is that this controller creates a single point of failure, if the controller fails, the entire network will be down. The controller also created a single point of attack, if an attacker managed to attack the controller, he can control the entire network. Table 2 shows a comparative analysis of SDN vs traditional networks.

Table 2. Comparative Analysis of SDN vs Traditional Networks.

Criterial	SDN	Traditional Networks
Global Network View	Central view at the controller.	Not possible.

Network Management	Easier with the help of the controller.	Changes need to be implemented separately for each device.
Maintenance Costs	Lower maintenance costs.	Higher maintenance costs.
Time for error handling and updating	Quick and easy because of the use of controllers.	Long and time-consuming.
Resource Utilization	High level of utilization.	Low level of utilization.
Security	The controller creates a single point of attack.	Low level vulnerable to security attacks.
Loop Avoidance	Native loop avoidance through the SDN controller.	Spanning-Tree Protocol (STP) must be used for loop avoidance.

5. Elimination of the Spanning-Tree Protocol in SDN

Spanning-Tree Protocol (STP) is a very important protocol in traditional networks to prevent the occurrence of any loops in layer-2 switches. STP is used to ensure that there is only one path that exists between any two points. SDN controllers can run STP function to stop flooding and to prevent broadcast storms on the network.

6. Comparative Analysis of Various SDN Controllers

In this section, we will evaluate the performance of several SDN controllers: OpenDaylight, ONOS, RYU, Floodlight, Floodlight-OF1.3, POX, and Trema.

Different metrics can be used to quantify the speed of transfer of data through a network among which latency, throughput, and topology discovery time are the most popular. Latency is the time taken for a packet to be transmitted through a network. Throughput is the amount of data being transmitted per second. Legacy Network Support is the support for backward compatibility with traditional routers and switches that contain their local control planes. SDN controllers that support network monitoring provide detailed accounting data with every flow by collecting these data from devices and provide them to the operator. Table 3 provides a comparison of the features of different SDN controllers.

Table 3. Comparison of the features of different SDN Controllers.

Criterial	OpenDaylight	ONOS	RYU	Floodlight	POX	Trema
Open Source	Yes	Yes	Yes	Yes	Yes	Yes
OpenFlow Support	V1.0	V1.0	V1.0 V1.2 V1.3	V1.0	V1.0	V1.0
Language Support	Java	Java	Python	Java	Python	C Runy
Legacy Network Support	Yes	Partial	Yes	Yes	No	No
Platform Support	Linux Mac Windows	Linux	Linux	Linux	Linux Mac Windows	Linux
Network Monitoring	Yes	Yes	Yes	Yes	Partial	Partial
Load Balancing	Yes	No	No	No	No	No

Figure 9 shows the results of measuring the average throughput for various controllers while Figure 10 shows the results for measuring the average latency for various controllers.

Figure 9. The average throughput for various controllers.

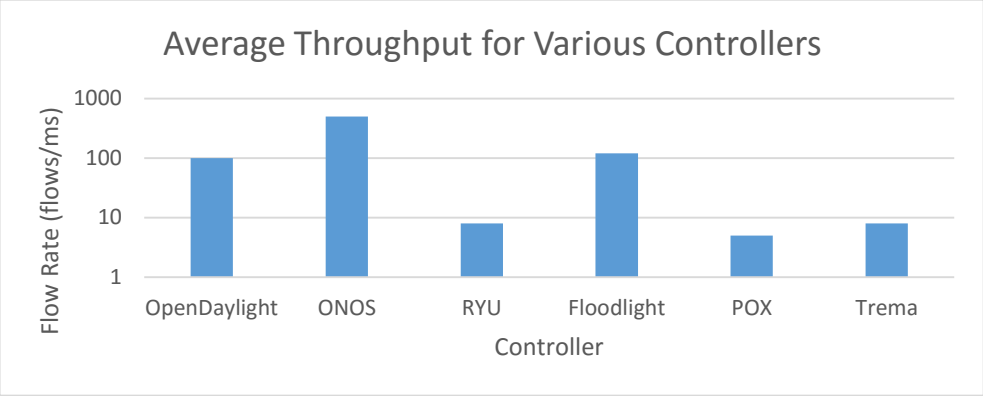
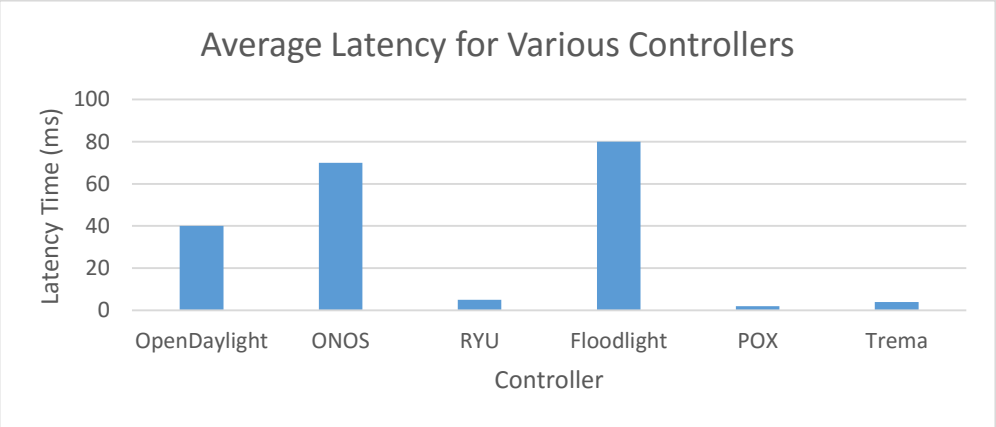


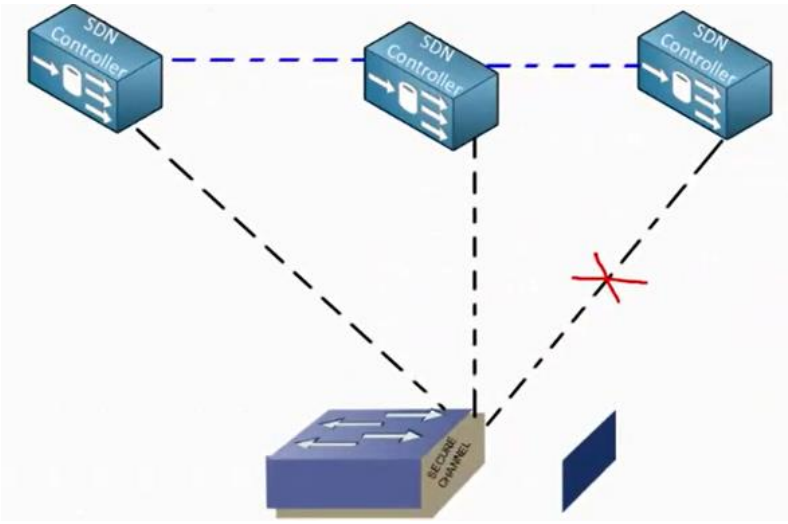
Figure 10. The average latency for various controllers.



7. Resilience Against Controller Failure

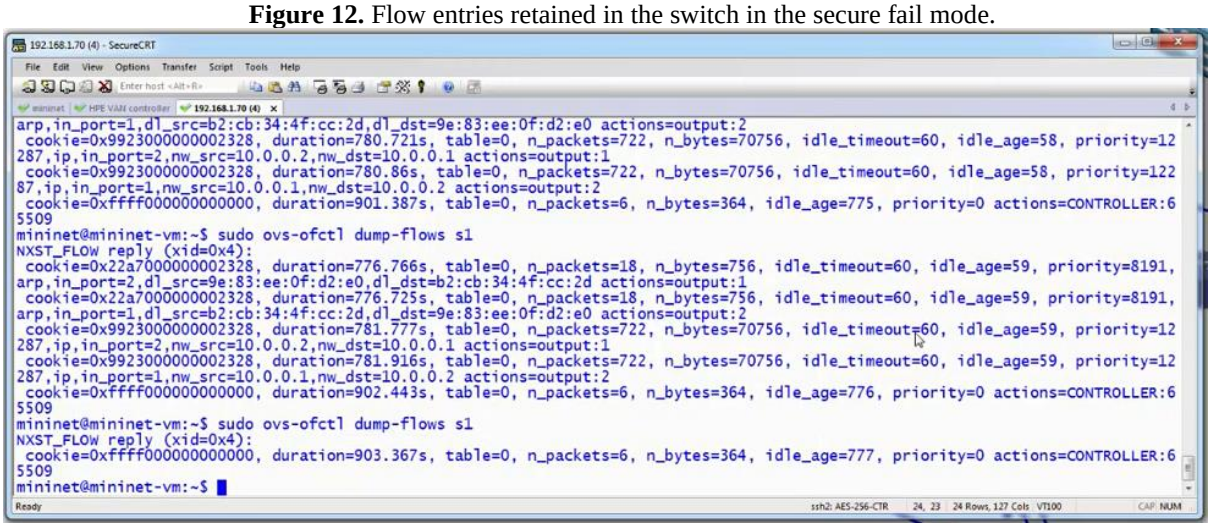
A secure channel session is maintained between the switch and the controller using TCP/TLS. In a scenario when the switch is connected to several redundant controllers as shown in Figure 11, only one controller at a time will be managing the switch to provide flow entries. If an interruption in the connection is detected with TCP/TLS time out, the switch sends controller status message to the other connected controllers, which in turn select among them in a fast way another controller to manage the switch. A controller status message is sent again when the failed controller session is up again.

Figure 11. Connection failure between the switch and the controller in a multiple redundant controller scenario.



In the case the connection with all controllers is interrupted, or in the case, there is only one controller and the connection is interrupted, the switch goes into standalone fail-safe more. If the switch is a hybrid switch, it activates it one control plane and operates like a traditional switch without controller support. If the switch is an open flow only switch, entries in the flow table are used to forward packets of similar source and destination address already having matching rules in the flow table, other flows are dropped and the switch fails to forward. However, due to the limited lifetime of entries in the flow table, which deletes a flow entry from the flow table after a certain amount of time passes with no flows matching that flow entry, the switch starts to drop those flows and fail to forward. The lifetime is found to be 60 seconds which can be increased to enhance the fail-safe behaviour of the switch. If the connection to the controller is resumed before 60 seconds, the switch resumes operation again with little impact of failure. Figure 12 shows flows entries in the switch. idle_timeout=60 indicates that the idle life time of flows in the switch is set to 60 seconds and idle_age is the time this flow is left with no traffic. In our experiment, until idle_age reaches 59, flows are retained in the switch, afterwards, flows are deleted and packets are dropped if the connection with the controller is not resumed with 60 seconds.

Figure 12. Flow entries retained in the switch in the secure fail mode.



8.Conclusion

In this research, we compared the performance of SDN networks to traditional networks and we showed how traditional networks suffer from several limitations such as data process forwarding is time-consuming, error-prone, and has high conversion times, besides, traditional networks limit automation, innovation and the ability to connect devices from different vendors together. SDN networks, on the other hand, have the advantages that the response to a link failure or node failure if faster, loop avoidance is simpler, management is simpler, less time consuming, money-saving, support for automation, innovation, and agility. We then used benchmarking tools to compare the performance of SDN controllers, we found that Multi-threaded controllers such as OpenDayLight, ONOS, and Floodlight perform better than centralized single-threaded controllers such as RYU, POX, and Trema in terms of latency and throughput. Variant features such as packet length and vSwitch buffer size may have an impact on the performance of the controller and these features need to be standardized for better quantification of controllers' performance. SDN is also resilient against controller failure in multiple redundant controllers' scenario and also in secure safe mode where flow entries are retained in the switch for an idle time of 60 seconds. SDN and Network Function Virtualization (NFV) will be deeply integrated in the fourth coming 5G technology and thus requires further investigations and studies.

References

1. Andrade, L., Borba, M., Ishimori, A., Farias, F., Cerqueira, E., & Abelém, A. (2016). On the benchmarking mainstream open software-defined networking controllers. Proceedings of the 9th Latin America Networking Conference, LANC 2016, 9–12. https://doi.org/10.1145/2998373.2998447
2. Basu, A., & Riecke, J. (2001). Stability issues in OSPF routing. 225–236. https://doi.org/10.1145/383059.383077
3. Correa Chica, J. C., Imbachi, J. C., & Botero Vega, J. F. (2020). Security in SDN: A comprehensive survey. Journal of Network and Computer Applications, 159(November 2019), 102595. https://doi.org/10.1016/j.jnca.2020.102595

4. Cui, L., Yu, F. R., & Yan, Q. (2016). When big data meets software-defined networking: SDN for big data and big data for SDN. IEEE Network, 30(1), 58–65. https://doi.org/10.1109/MNET.2016.7389832
5. Dai, B., Xu, G., Huang, B., Qin, P., & Xu, Y. (2017). Enabling network innovation in data center networks with software defined networking: A survey. Journal of Network and Computer Applications, 94(July), 33–49. https://doi.org/10.1016/j.jnca.2017.07.004
6. Dixon, I. C. K., Us, T. X., Felter, W. M., Us, T. X., Jose, S., Us, C. A., & Shaikh, A. A. (2016). (12) United States Patent (10) Patent No .: 2(12).
7. Gracia-Tinedo, R., Sampé, J., París, G., Sánchez-Artigas, M., García-López, P., & Moatti, Y. (2019). Software-defined object storage in multi-tenant environments. Future Generation Computer Systems, 99, 54–72. https://doi.org/10.1016/j.future.2019.03.020
8. Greenberg, A. (2015). SDN for the Cloud Road to SDN, Keynote in the 2015 ACM Conference on Special Interest Group on Data Communication. ACM Conference on Special Interest Group on Data Communication.
9. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). NOX: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3), 105–110. https://doi.org/10.1145/1384609.1384625
10. Haggag, A. (2019). Network Optimization for Improved Performance and Speed for SDN and Security Analysis of SDN Vulnerabilities. International Journal of Computer Networks and Communications Security, 7(July), 83–90.
11. Hoang, D. B., & Pham, M. (2015). On software-defined networking and the design of SDN controllers. 2015 International Conference on the Network of the Future, NOF 2015. https://doi.org/10.1109/NOF.2015.7333307
12. Hu, F., Hao, Q., & Bao, K. (2014). A survey on software-defined network and OpenFlow: From concept to implementation. IEEE Communications Surveys and Tutorials, 16(4), 2181–2206. https://doi.org/10.1109/COMST.2014.2326417
13. Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., & Tran-Gia, P. (2011). Modeling and performance evaluation of an OpenFlow architecture. Proceedings of the 2011 23rd International Teletraffic Congress, ITC 2011, 1–7.
14. Li, D., Hong, P., Xue, K., & Pei, J. (2019). Virtual network function placement and resource optimization in NFV and edge computing enabled networks. Computer Networks, 152, 12–24. https://doi.org/10.1016/j.comnet.2019.01.036
15. Liu, T. (2014). Implementing Open flow switch using FPGA based platform. June, 1–140.
16. Pham, M., & Hoang, D. B. (2016). SDN applications - The intent-based Northbound Interface realisation for extended applications. IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conference and Workshops: Software-Defined Infrastructure for Networks, Clouds, IoT and Services, 372–377. https://doi.org/10.1109/NETSOFT.2016.7502469
17. Shankar, G. H. (2016). OFNet. OFNet-Quick User Guide.[Online]. Available: Http://Sdninsights. Org/.[Accessed: 05-Jun-2018].
18. Sherwood, R., Gibb, G., Yap, K., Appenzeller, G., Casado, M., Mckeown, N., & Parulkar, G. (2009). FlowVisor: A Network Virtualization Layer. Network, 15. https://doi.org/10.1007/s13398-014-0173-7.2
19. Sherwood, R., & Kok-Kiong, Y. (2010). CBench: An openflow controller benchmark tool.
20. World, R., & Emulation, N. (n.d.). Real World Network Emulation Testing and Performance Benchmarking of SDN Controllers Software based solution emulates large scale SDN networks with thousands of nodes and millions of flows Lower investment on test infrastructure REPORTS & STATISTICS Ben. 1–4.