International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-9 Issue-2, July 2020

To eliminate the threat of a Single Point of Failure in the SDN by using the Multiple Controllers

Deepjyot Kaur Ryait, Manmohan Sharma

Abstract: ***The Software Defined Network (SDN) provides an innovative paradigm for networking, which improve the programmability and flexibility of the network. Due to the separation between the control and data plane, all the control logic transfer to the controller. In SDN, the controller, which provides a global view of the whole network. That is why it acts as the "Network Brain" of the network. Because the controller has the capability to configure or reconfigure the forwarding devices by customizing their policies in a dynamic manner. Thus, the controller provides a centralized logical view of the entire network. Therefore, all manipulation and implementation in the network are control by the single controller in the SDN, which increases the maximum chance of a single point of failure (SPOF) in the network. As a consequence, it collapses the entire network. Therefore, a fault tolerance mechanism is required which reduce single point of failure in the network by using multiple controllers. As a significance, this mechanism also increases the scalability, reliability, and high availability of services in the network. The three different roles of multiple controllers are equal, master and slave exist in the SDN. In the simulation, the Ryu SDN controller and Mininet tool are utilized. During the simulation to analysis, what is happen when a single point of failure (SPOF) occur in the network and how to use the different roles of the multiple controllers (such as equal, master and slave) which reduces the threat of single point of failure in SDN network.***

Keywords: SDN, SPOF, Mininet, OpenFlow, Ryu.

I. INTRODUCTION

Computer Network has an important role in the communication system. But the lack of programmability and flexibility in traditional networks become the cause of root for promotion in the network field. The main reason behind this is a strong bound existence between the functional components. Thus, Software Defined Network (SDN) provides a novel paradigm architecture of the network by decouple the control plane from data plane. Due to this separation, SDN offers programmability, adjustable, and dynamic reconfiguration of the networking devices.

Thus, Software Defined Networks enhance the innovation in the network field which copes up the requirements of the network user on demand. Thus, Software Defined Network also offers the flexibility, centralized view control, decrease in the complexity and the cost of network systems.

It increases the efficiency of network by improving the network control which enables the network providers to respond to the changing or varying the business requirements swiftly [1-7]. Currently, many industries support the Software Defined Networks paradigm like Microsoft, Google, Cisco, Facebook, HP, IBM, Samsung, VMware, Juniper, etc.

II. OVERVIEW OF SDN ARCHITECTURE

The Software Defined Network paradigm divided into the three different planes are Data Plane, Control Plane, and Application Plane as shown in Figure 1. All the forwarding devices mainly exist in the data plane. Due to this separation, the data plane act as a simpler forwarding element whose behaviour is dedicated by the controller. Thus, it is also known as a forwarding plane. The control plane usually called the controller, which provides a global view of the entire network. That is why it acts as the "Network Brain" of the network [1-5]. The controller has the capability to configure or reconfigure the forwarding devices by customizing their policies in a dynamic manner.

The control plane act as an intermediate between the application and data plane. In application plane, various network applications are implemented to control the logic of the network domain. These applications run on the top of the controller. The communication between the application and control plane is possible through the northbound APIs such as the REST API's and the communication between the data plane and the control plane is possible through the southbound APIs such as the OpenFlow protocol.

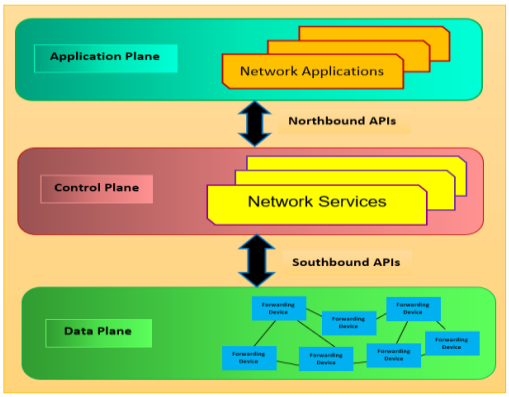


Figure 1: Software Defined Network Paradigm

Manuscript received on May 25, 2020.
Revised Manuscript received on June 29, 2020.
Manuscript published on July 30, 2020.
***** Correspondence Author
Deepjyot Kaur Ryait*, School of Computer Applications, Lovely Professional University, Phagwara, India. Email: djryait@gmail.com
Dr. Manmohan Sharma, School of Computer Applications, Lovely Professional University, Phagwara, India. E-mail: manmohan.sharama71@gmail.com

III. OPENFLOW PROTOCOL

In SDN, the most famous southbound interface is the OpenFlow protocol, which makes communication between the control and data plane. The various OpenFlow versions available at present are OpenFlow 1.0, OpenFlow 1.1, OpenFlow 1.2, OpenFlow 1.3, OpenFlow 1.4 and OpenFlow 1.5. But the most widely supported version is OpenFlow 1.3. Both switch and controller must have the same version of the OpenFlow during establishing the communication channel between them. The OpenFlow 1.1 version, has two major changes are occurring such as the pipeline of the multiple flow tables and a group table which is used during the packets processing. For this reason, OpenFlow V1.0 and OpenFlow V1.1 are not compactable to each other. The OpenFlow switch mainly consists of two components are [1,2,5,19]:

1. **Secure Channel**: It is used to establish a connection between the controller and the switch by using an encrypted channel (usually via SSL and TLS).
2. **Flow Table**: Number of forwarding rules (i.e. flow entries) are exist in the flow table. These forwarding rules are used for processing the incoming packets.

At least one flow table must be contained in each OpenFlow switch. The number of flow entries is existing in the flow table which is organized by priority order. Each flow entry has mainly three fields/ components are given below (as shown in Figure 2):

- **Matching/Header Field**: To matching the header information of the incoming packet against the source and destination address, port address, IP address/MAC address and VLAN ID.
- **Actions/Instructions**: It specifies corresponding action takes place for the matching packet such as the forward the packet, drop the packet, modify the packet, etc.
- **Stats**: To include statistical information about the flows such as the number of received packets and bytes, as well as the duration of the flow.

The controller has responsibility for installing and manipulate the flow rules in the switches of the data plane. Thus, the controller has privileges to the insert, delete, and modify of the flow entries in the flow table, which instructs the switches how to forward the packet in the data plane. So, the controller can easily manipulate the forwarding behaviour of the switches.

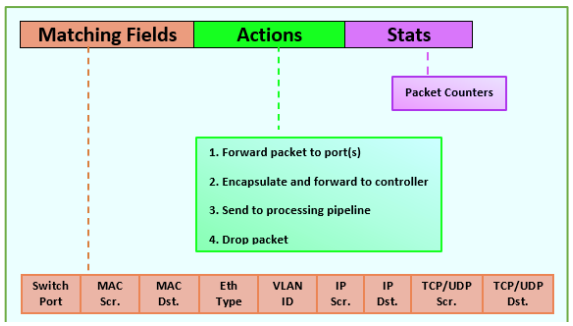


Figure 2: Format of the Flow Table

For this propose, the OpenFlow protocol supports various type of messages which define how to coordinate the forwarding devices in the data plane. These messages are subdivided into three types of the category are:

- **Controller-to-Switch**: These messages are originated by the controller to directly examine or manage the state of the switch such as PACKET-OUT, FLOW-MOD, ROLE-REQUEST, etc.
- **Asynchronous**: These messages are initiated by the switch to inform the controller about the network events and changes in the switch state like PACKET-IN, PORT-STATUS.
- **Symmetric:** Symmetric messages are introduced either by the controller or switch to send without any solicitation such as HELLO, ECHO message.

In SDN, the controller can used two different way to implement flow rules in the flow table are:

1. Reactive Flows
2. Proactive Flows

- **Reactive Flows**: In this method, when a host sends a new packet in the network then the switch matches its header field to the corresponding of flow entries in the table. If the header of packet is matched then corresponding action is executed. If the header of the packet is not matched then switch send PACKET-IN message to the controller. Then the controller installs new or modifies flow rules in table by using FLOW-MOD or PACKET-OUT message to the switches of the networks. After that switches can easily forward the packets to the desire destination [1-19].
- **Proactive Flows**: In this method, the OpenFlow controller will install the flow tables ahead of time for all the traffic matches in the switch. So never occur PACKET-IN event in this method because the controller can populate the flow table before the packet arrived. Thus, proactive flow eliminates latency which is induced due to the involvement of the controller in the PACKET-IN message. In the proactive method, the value of the action field is FLOOD.

IV. IMPORTANCE OF THE CONTROLLER IN SDN

In SDN, the controller, which provides a global view of the entire network. That is why it acts as the "***Network Brain***" of the network. The controller has the capability to configure or reconfigure the forwarding devices by customizing their policies in a dynamic manner. The control plane act as an intermediate between the application plane and the data plane. The controller has the facility of the network management and also has capability to solve network related problems by providing the logically centralized view of the entire network. Thus, when defining network policies, the developer has no need to maintain information about the low-level details of the data distribution among the forwarding elements [10-19]. Due to this reason,

The controller considers a critical component in the Software Defined Network for the following reasons:

- ✓ The controller provides a logically centralized view of the whole network.
- ✓ The controller has the ability to control the network traffic of the data plane.
- ✓ How to maintain and update the topology information in the network.
- ✓ How to install the flow rules in forwarding elements.
- ✓ How to reduce the complexity of network management through programmability.
- ✓ The controller decides to implements flow rules either using a reactive or proactive manner.
- ✓ When a large number of traffic handle as comparing its capacity then it increases the latency of the network and decreases the performance of the network.

For the above reasons, the single controller is not feasible in the Software Defined Network. Because it increases the maximum chance of failure in the network. In a case when the controller is failed for any reason, then the entire network becomes halt. This situation is not desirable in the communication network. To overcome this problem, the fault tolerance mechanism is required which eliminates the risk of single point of failure (SPOF) in the Software Defined Networks [9].

A fault tolerance mechanism empowers the system to continue its functionality even the presence of failure in its components. That is why propose to use the multiple controllers in the Software Defined Network. As a consequence, the multiple controllers can reduce a single point of failure in SDN as shown in Figure 3.

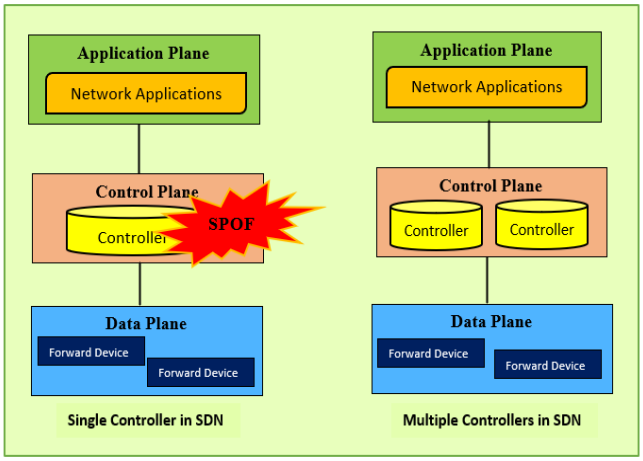


Figure 3: Single Controller v/s Multiple Controllers in SDN

V. ROLE OF MULTIPLE CONTROLLERS IN THE SOFTWARE DEFINED NETWORK

In Software Defined Network, to eliminate the threat of single point of failure then promote the use of multiple controllers in the network. The OpenFlow specification supports multiple controllers' environment, in which the controller has anyone role of the following. The three types of controller's roles are (shown in Figure 4):

1. **Equal Role**: In the equal role all the controllers configured in the switch have full control to update or modify the flows. The switch must send the PACKET_IN message to all the controllers. And also switch process the PACKET_OUT, FLOW_MOD, etc. from all the controllers.
2. **Master Role**: The master controller has the responsibility for managing the switches of the data plane. These switches will send the control message to the master controller only.
3. **Slave Role**: The slave controller plays the backup role for the master controller. It also receives the HELLO and KEEPALIVE messages. But it cannot send and receive the control messages.

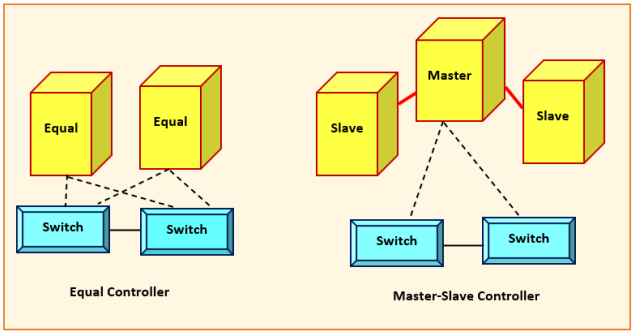


Figure 4: Role of Multiple Controllers in the SDN

Each switch can have a maximum one master controller, but it can have multiple equal or slave controllers. Thus, the multiple controllers improve the reliability of the network in case of the master controller failure [9]. Then other controller sends the ROLE_REQUEST (as the ROLE_MASTER) message to the switch for changing their state(role). On receiving this message then switch send back ROLE_REPLY message to the controller. And the other controllers send the role as the SLAVE. The switch will communicate with the master controller [14-19].

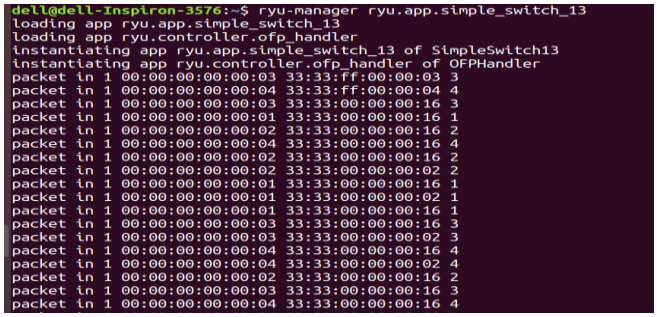
VI. SIMULATION TESTBED AND RESULTS

To set up the SDN test environment with the Ryu SDN controller require to install the following tools are [20-24]: -

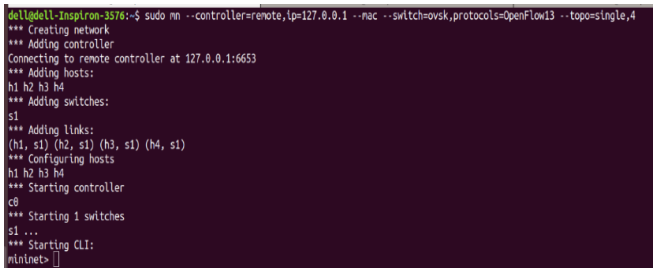
- Operating System: - Ubuntu 18.04 Desktop
- Test Bed/ Simulator: - Mininet
- Controller: - Ryu
- Switch: - Openvswitch
- Language: - Python

The simulation is conducted to test how the reactive and proactive flow implemented by the controller. After installing all required tools then open the three terminals to check the behaviour of the reactive flow rules implemented by the controller in the SDN by using the following steps (in Figure 5):

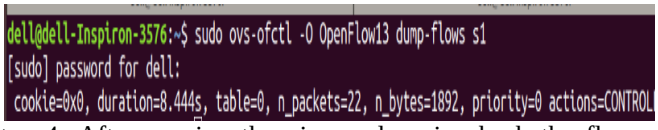
Step 1: Run the Ryu Controller in Terminal 1 shown in Figure 5(a).



Step 2: Run the single topology using the Mininet in Terminal 2 shown in Figure 5(b).



Step 3: Run the following command in Terminal 3 to check to the Table-Miss Entry whose table-id and priority is always zero shown in Figure 5(c).



Step 4: After running the ping and again check the flow entries in Terminal 3 shown in Figure 5(d).

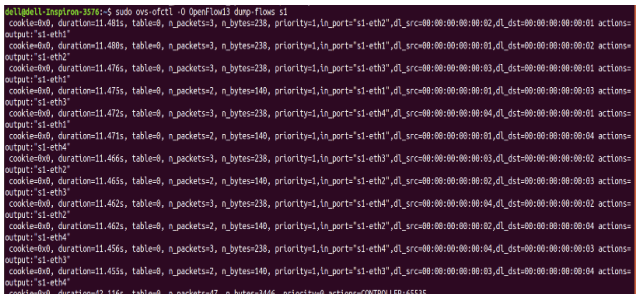


Figure 5: Reactive Flow Rules in the SDN

Similar, the controller implements the proactive flow rules in the SDN, but the value of action filed is equal to FLOOD as shown in Figure 6:

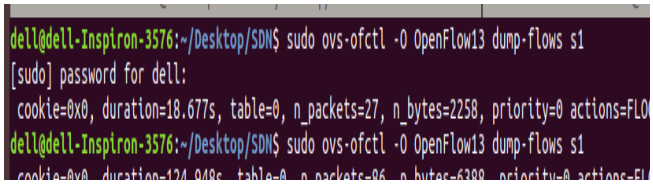


Figure 6: Proactive Flow Rules in the SDN

In the OpenFlow protocol, the TCAM memory is more preferable to hold the flow table because it provides flexibility and efficiency in the term of the matching capabilities, but it is very expensive and small in size. So, the TCAM memory is insufficient to hold large number of flow entries in a flow table simultaneously [1,5,13]. Thus, the OpenFlow provides a flow timeout facility which enables the flows will be expired or removed after a certain time. For this purpose, two fields available are:

- Idle Timeout: If the flow is idle for the specified time then flow will be expired.
- Hard Timeout: The flow rule must be expired within the specified time limit (in the number of seconds). Whether or not packets are matching (or hitting) the entries.

By default, the value of timeout is zero in both fields, it means the flows are permanent. It will never be expired as shown below in Figure 7:

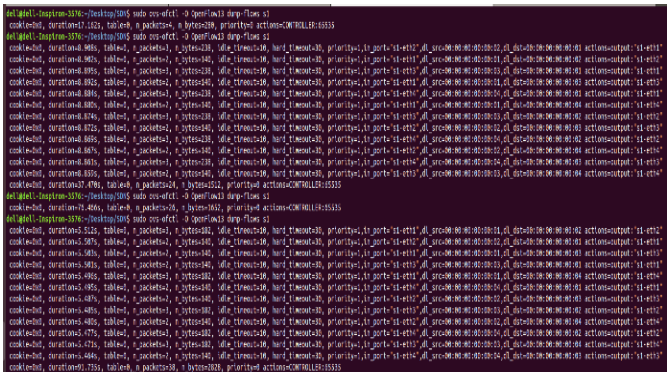


Figure 7: Idle and Hard Timeout fields in the Flow Rules

The SDN controller has also the responsibility to maintain the entire topology information. For this propose the controller generate the LLDP (Link Layer Discovery Protocol) packet to determine (or discover) the topology information about each switch, port, and link in the network as shown below in Figure 8.

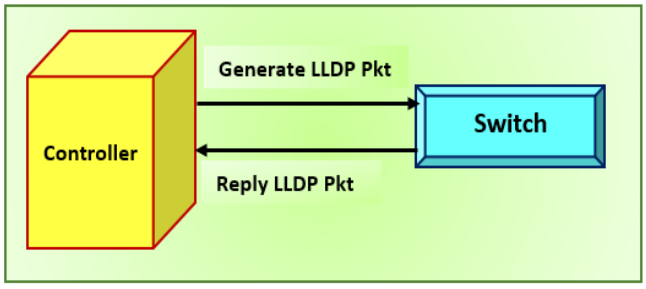
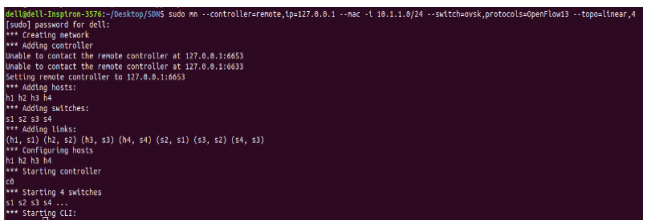


Figure 8: Propose of LLDP Packet between the Controller and Switch

The Ryu controller used –observe-links flag to get topology information of the network. Now open two terminals, in the first terminal run linear topology in Mininet shown in Figure 9 (a) and in the second terminal run Ryu controller with –observe-links and topology-discovery file which prints the Topology Information on the Ryu console a shown below in Figure 9 (b).



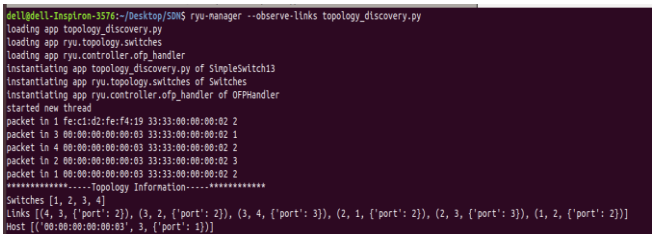
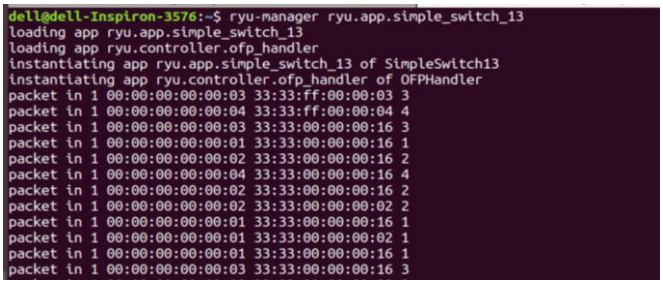


Figure 9: Display Topology Information in SDN

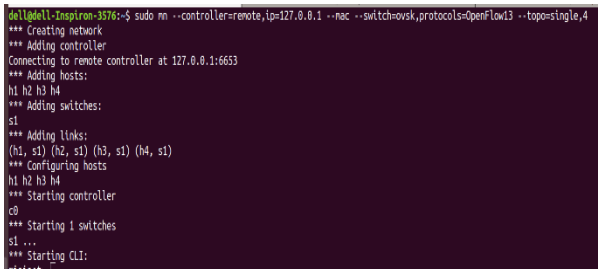
In SDN, the single controller is not feasible because it increases the maximum chance of a single point of failure in the network. When the single controller becomes failed for any reason then the entire network becomes unreachable in Figure 10 (c).

The simulation of a single controller in the network as follows the following steps:

Step 1: Run the Ryu controller in Terminal 1 with Ryu App simple_switch_13 as shown below in Figure 10 (a).



Step 2: Run the Single topology in Mininet in Terminal 2 and start ping command between hosts h1 and h2 as shown in Figure 10 (b).



Step 3: After sometime kill the Ryu Controller and check the behaviour of the ping command as shown in Figure 10 (c).

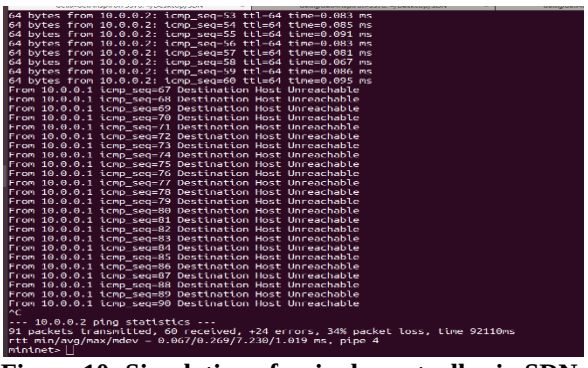


Figure 10: Simulation of a single controller in SDN

When the controller failed then the ping command does not reachable the destination host. Therefore, the packet loss percentage increases gradually in the network. The following graph show number of packets transmitted, received packets, and packet loss when a single controller fails in Figure 11.

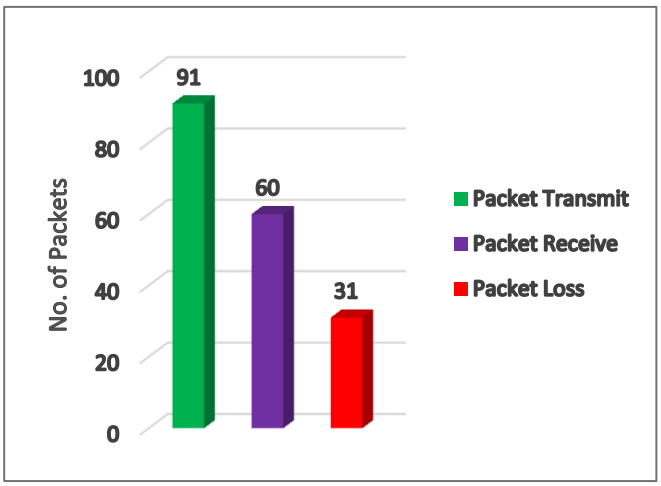


Figure 11: Representation of packets in term of Transmit, Receive and Loss

The below graph shows the three different scenarios of the number of packets transmitted, packet receive and packet loss when a single controller failed in the network in Figure 12.

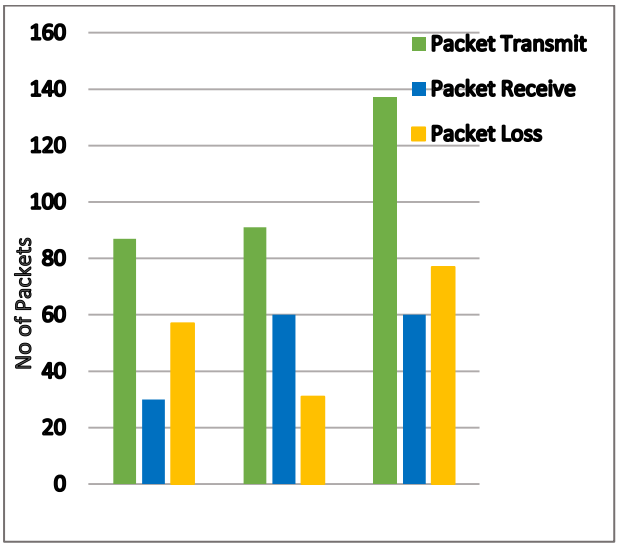


Figure 12: Representation of three different scenarios in term of Transmit, Receive and Loss Packets

To overcome a single point of failure in SDN, then propose to use the multiple controllers in the Software Defined Network either equal controllers or master-slave controllers. In the simulation of equal controller configuration run two remote controllers in the network with different listen port-number as shown below Figure 13(a) and Figure 13 (b).

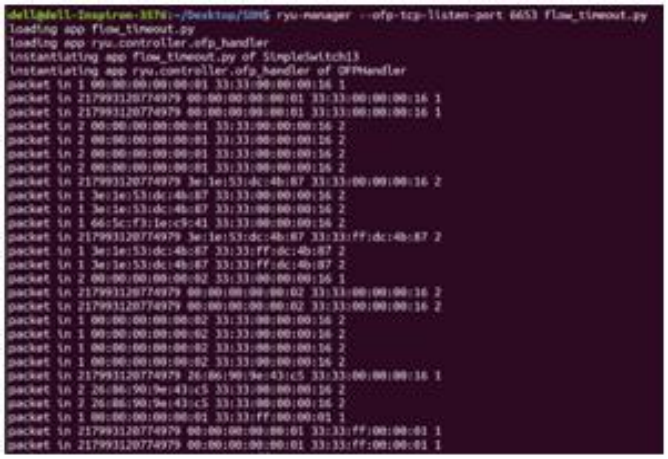


Figure 13: Simulation of the Equal Controllers in SDN

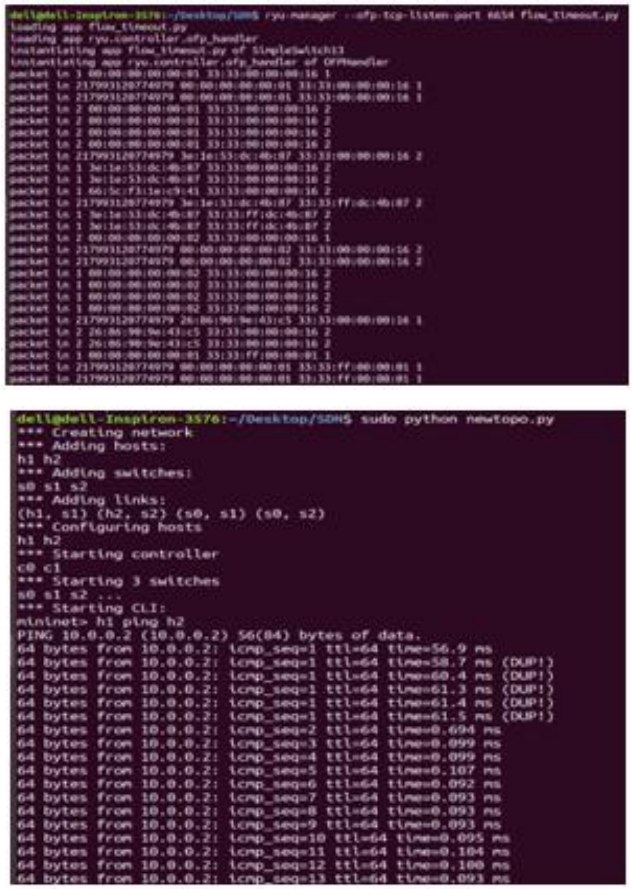


Figure 14: Equal Controllers Generate Duplicate Packets in the Network

In Equal controller configuration, generate duplicate packets because every controller has privileges to process the PACKET_IN message individually and respond to it. Therefore, it spawns the duplicate packets in the network shown in Figure 14. But in case any controller fails in the network, then there is no packet loss occur in the network because of another controller continuous process without any problem.

The below graph (in Figure 15) shows the two different scenarios of the number of packets transmitted, packet receive, and packet duplicate in the equal controller configuration.

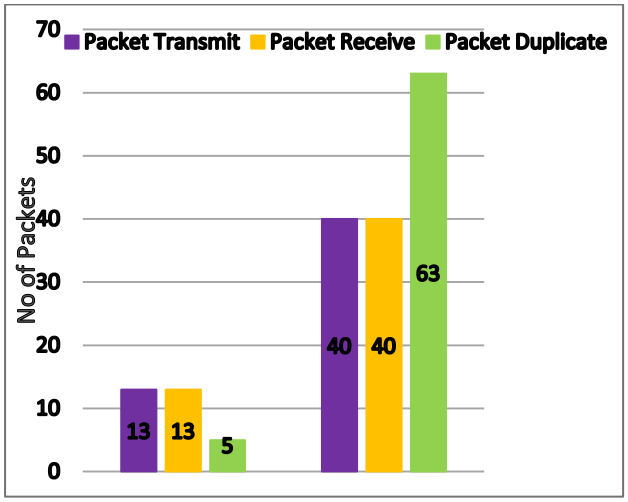


Figure 15: Representation of two different scenarios in term of Transmit, Receive and Duplicate Packets

In multiple controllers' environment also have the master-slave configuration, in which one controller acts as the master and other controller behaviour as slave controllers in the networks. In the master-slave configuration, there are no duplicate packets generate as compared to the equal controllers. The simulation of the master-slave configuration, the master controller uses the listen port-number 6653 in Figure 16 (a) and the slave controller listen port-number 6654 in Figure 16 (b). After running the topology then the host "h1" ping to host "h2" without generating the duplicate packets in the network as shown in Figure 16 (c).

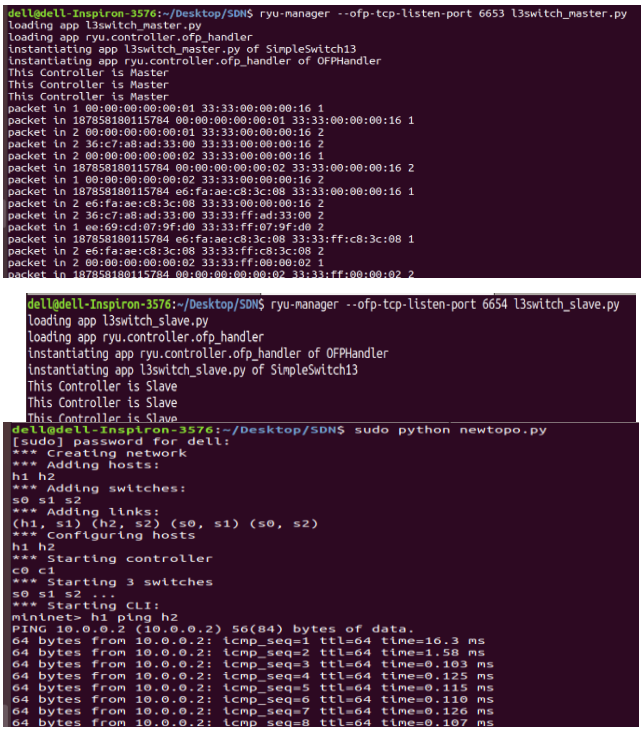


Figure 16: Simulation of the Master-Slave Controllers in SDN

After conducting the simulation on the different roles of controllers in the Software Defined Networks (SDN) by using the Ryu controller and the Mininet tools and results are summarized in a tabular form as shown in Figure 17.

Comparison Between Different Role of Controllers in SDN	Single Controller Environment in SDN	Multiple Controllers Environment in SDN	
		Equal Controller	Master-Slave Controller
Threat of SPOF in Network	✓	×	×
Packet Loss after SPOF	✓	×	×
Duplicate Packet Generate	×	✓	×

Figure 17: Simulation Result on various SDN Controllers

VII. CONCLUSION

Software Defined Network provides a novel paradigm of networking by decoupling the control plane from the data plane. Due to this separation, the controller provides a logically centralized view of the entire network. Therefore, it increases the maximum chance of single point of failure in the network. To eliminate the risk of a SPOF in the network by using the multiple controllers in the SDN. In the simulation, analysis when a single controller failed in the network then the percentage of packet loss increases. To overcome this problem, use the multiple controllers in the SDN. In simulation when using the equal controllers in the network, then there is no packet loss occur but the duplicate packets are generated by the controllers in the network. To avoid duplicate packets in the network then use the master-slave configuration in the multiple controllers. After the simulation, the summarized results shown in Figure 17. But the data synchronous problem has occurred between the master and slave controllers. Thus, in the future try to maintain data synchronization between the master-slave controllers to enhance the availability of the network.

REFERENCES

1. D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.
2. W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," Open Access Future Internet, vol. 6, no. 2, pp. 302-336, 2014.
3. Y. Yu, X. Li, X. Leng, L. Song, K. Bu, J. Yang, Y. Chen, L. Zhang, K. Cheng and X. Xiao, "Fault Management in Software-Defined Networking: A Survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp 349-392, 2018.
4. C. M. Duran, E. A. Leal and J. F. Botero, "Improving fault tolerance in critical networks through OpenFlow," in IEEE Colombian Conference on Communications and Computing (COLCOM), IEEE, 2017.
5. A. Malik, B. Aziz, A. Al-Haj and M. Adda, "Software-Defined Networks: A Walkthrough Guide From Occurrence To Data Plane Fault Tolerance," Open Access, pp. 1-26, 2019.
6. J. Chen, J. Chen, F. Xu, M. Yin and W. Zhang, "When Software Defined Networks Meet Fault Tolerance: A Survey," G. Wang et al. (Eds): International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Part III, vol. 9530, pp. 351-368, 2015.
7. M. R. Parsaei, S. H. Khalilian and R. Javidan, "A Comparative Study on Fault Tolerance Methods in IP Networks versus Software Defined Networks," International Academic Journal of Science and Engineering, vol. 3, no. 4, pp. 146-154, 2016.
8. B. Isong, I. Mathebula and N. Dladlu, "SDN-SDWSN Controller Fault Tolerance Framework for Small to Medium Sized Networks," in the 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE Computer Society, pp. 43-51, 2018.
9. L. Sidki, Y. Ben-Shimol and A. Sadovski, "Fault Tolerant Mechanisms for SDN Controllers," in IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, pp. 1-6, 2016.
10. Y. Zhang, L. Cui, W. Wang and Y. Zhang, "A Survey on Software Defined Networking with Multiple Controllers," Journal of Network and Computer Applications (Elsevier), pp. 1-58, 2017.
11. D. Gopi, S. Cheng and R. Huck, "Comparative Analysis of SDN and Conventional Networks using Routing Protocols," in IEEE International Conference on Computer, Information and Telecommunication Systems (CITS), IEEE, 2017.
12. A. J. Gonzalez, G. Nencioni, B. E. Helvik and A. Kamisinski, "A Fault-689+Tolerant and Consistent SDN Controller," in IEEE Global Communications Conference (GLOBECOM), IEEE, 2016.
13. M. Z. Abdullah, N. A. Al-awad and F. W. Hussein, "Performance Evaluation and Comparison of Software Defined Controllers," International Journal of Scientific Engineering and Science, vol. 2, no. 11, pp. 45-50, 2018.
14. Y. E. Oktian, S. Lee, H. Lee and J. Lam, "Distributed SDN controller system: A survey on design choice," Elsevier Computer Networks, vol. 121, pp. 100-111, 2017.
15. M. Karakus and A. Durresi, "A survey: Control Plane Scalability Issues and approaches in Software-Defined Networking (SDN)," Elsevier Computer Networks, vol. 112, pp. 279-293, 2017.
16. F. Bannour, S. Souihi and A. Mellouk, "Distributed SDN Control: Survey, Taxonomy and Challenges," IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 333-354, 2017.
17. M. Paliwal, D. Shrimankar and O. Tembhurne, "Controllers in SDN: A Review Report," IEEE Access, vol. 6, pp. 36256-36270, 2018.
18. A. Mahjoubi, O. Zeynalpour, B. Eslami and N. Yazdani, "LBFT: Load Balancing and Fault Tolerance in distributed controllers," in 2019 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2019.
19. A. U. Rehman, R. L. Aguiar and J. P. Barraca, "Fault- Tolerance in the Scope of Software-Defined Networking (SDN)," IEEE Access, vol. 7, pp. 1-18, 2019.
20. S. Asadollahi, B. Goswami and M. Sameer, "Ryu Controller's Scalability Experiment on Software Defined Networks," in 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), IEEE, 2018.
21. P. Dutta and R. Chatterjee, "A Novel Solution for Controller Based Software Defined Network (SDN)," in Communications in Computer and Information Science, Springer Singapore, 2018.
22. "Ryu SDN Controller," [Online]. Available: https://osrg.github.io/ryu/. [Accessed: 20- Apr-2020].
23. "Getting-Started-Ryu 4.30 documentation," [Online]. Available: https://ryu.readthedocs.io/en/latest/getting_started.html. [Accessed: 20- Apr-2020].
24. "Mininet," [Online]. Available: http://mininet.org/. [Accessed: 20-Apr-2020].

AUTHORS PROFILE

Deepjyot Kaur Ryait is a researcher scholar in the Department of School of Computer Applications at Lovely Professional University, Phagwara, India. She has done her Post Graduation in the M.Sc. (Information Technology) from Ramgarhia Girls College, Ludhiana in 2007 which is affiliated by the Panjab University, Chandigarh. Prior to that, she has done her undergraduate degree in the Bachelor of Computer Science and Applications from Khalsa College for Women, Ludhiana in 2005 which is affiliated by the Panjab University,

Chandigarh. She has also successfully completed her National Programme on Technology Enhanced Learning (NPTEL) Online Certificate on "Introduction to Internet of Things" with 100% and was among 1% of Topper with Gold Medal in Jan-Apr 2019. Along with that, she is also working as an Assistant Professor in the Computer Science and Applications Department. She has over 13 years' experience. Her current research area of interest includes the new network paradigm of architecture which is mainly based on Software Defined Networks (SDN) and the Internet of Things (IoT).

Dr. Manmohan Sharma serving as Associate Professor in School of Computer Applications, Lovely Professional University, Punjab, INDIA has a vast experience of more than 20 years in the field of academics, research and administration with different Universities and Institutions of repute such as Dr. B.R. Ambedkar University, Mangalayatan University etc. Dr. Sharma has been awarded with his Doctorate degree from Dr. B.R. Ambedkar University, Agra in 2014 in the field of Wireless Mobile Networks. His areas of interest include Wireless Mobile Networks, Adhoc Networks, Mobile Cloud Computing, Recommender Systems, Data Science and Machine Learning etc. A large number of research papers authored and co-authored, published in International or National journals of repute and conference proceedings comes under his credits. He is currently supervising six doctoral theses. Three M.Phil. degrees has already awarded under his supervision. He has guided more than 1000 PG and UG projects during his service period under the aegis of various Universities and Institutions. He worked as reviewer of many conference papers and member of the technical program committees for several technical conferences. He is member of various professional/technical Societies including Computer Society of India (CSI), Association of Computing Machines (ACM), Cloud Computing Community of IEEE, Network Professional Association (NPA), International Association of Computer Science and Information Technology (IACSIT), and Computer Science Teachers Association (CSTA).