International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

Analysis to Prevent Case Cading Failure for Controller Placement in Software-Defined Networking and its Implementation Using Dynamic Switch Assignment

Ritesh Jain[1]
Ph. D. Research Scholar
Department of Computer Science and Engineering
Dr. A.P.J. Abdul Kalam University, Indore (M.P.) - 452010, India
Email: rit.rit1@gmail.com

Dr. Pradnya Ashish Vikhar[2]
Research Supervisor
Department of Computer Science and Engineering
Dr. A.P.J. Abdul Kalam University , Indore (M.P.) - 452010, India
Email: pradnyav123@gmail.com

Abstract: The control and data planes are decoupled in programming portrayed getting sorted out (SDN), which enables the two planes to progress unreservedly, and accomplishes various advantages like high flexibility, programmability, and quick execution of new association shows. Regardless, to improve the versatility of the control plane as of now, some control functionalities are added to the data plane, which is probably going to influence on the agreement of the data plane. The basic trial of adding control functionalities to the data plane is to track down some sort of congruity between the agreement of the data plane and the versatility of the control plane. We propose some fundamental guidelines that both control and data planes should adjust to, considering the formative example of SDN. Moreover, we receive two methodologies for reference according to the principles, seen from the control messages in OpenFlow-based SDN. Our evaluations display that the systems can keep up the distortion of the data plane and improve the flexibility of the control plane.

Keywords : *OpenFlow, SDN, Designing, Control Functionalities, Control message*

I. INTRODUCTION

Present PC associations are utilizing enormous and complex organizations, there are numerous supplies engaged with PC networks like switches and switches, firewalls, network address interpreters, interruption location frameworks. In current situation we can't store the entirety of our data on the neighborhood frameworks and because of the expanding volume of the information organizations have moved towards another idea which is known as a server farm.

Server farm dependability in reality is emphatically relies upon the association running the server farm, not simply on the plan. The warmth produced by all hardware is eliminated by datacenter cooling frameworks. There should be some progression of circle framework in a cooling framework for eliminating the warmth, each time the order brings a cool medium that heats up from some warmth trade and again cooled back in some way or another.

1.1 Software Defined Networking (SDN):

Control plane is the capacities in the organization that controls conduct of the organization. Ordinarily the control plane is launched as a solitary, significant level programming regulator. It is organization's cerebrum and that is basically the thing is controlling the conduct of the organization. Information plane is the capacities in the organizations that are liable for sending the traffic. Control Channel is the correspondence channel over which a SDN regulator speaks with the fundamental organization switches.

1.2 Tools and Platform Studied:

At the point when we run Floodlight, tasks of both the northward and southward APIs from the regulator gets

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

dynamic. Any application can collaborate with the regulator by sending an http REST order. Then again, at the outbound, the supplier module of Floodlight will begin tuning in on the OpenFlow-determined TCP-port for associations from the OpenFlow switches. Floodlight, as of now upholds OpenFlow 1.0.

With an extensible Java improvement climate, and undertaking grade center motor, Floodlight is both a simple to utilize and powerful SDN regulator. Mininet executes python in the engine. Mininet runs genuine part, switch, it can run genuine programming and it can run genuine application code on a solitary machine. Numerous OpenFlow highlights are as of now inherent so it is valuable in creating, conveying and sharing different things which we do in Mininet programming climate.

The detachment of information plane and control plane permits network administrator to control network conduct from an incorporated single undeniable level control program. In the current engineering average organization gadgets are switches, a control plane and different highlights. The Management plane uses a few conventions like straightforward organization the board convention, Telnet, HTTP, secure HTTP and SSH. With the assistance of these orders switches discovers the organization geography and chooses the conduct of physical and virtual switches, and everything relies upon the solicitations of uses from the northward APIs. The early advantages of SDN are to a great extent going to come from the utilization of organization virtualization, which considers more unique organization division and use.

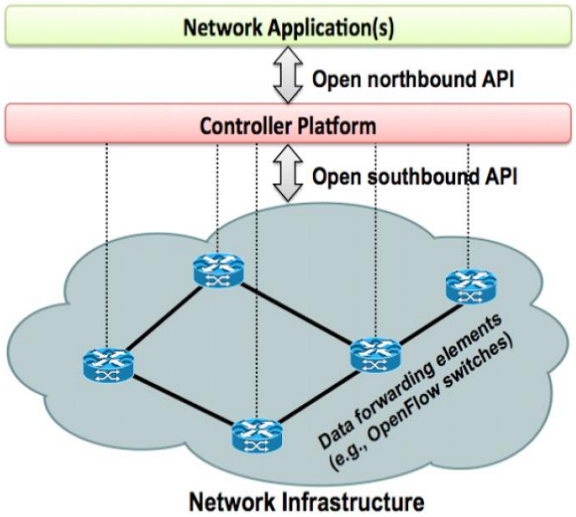


Figure 1: Simplified View of SDN Architecture

What's more, parcels are sent by the switches dependent on the regulator directions. Programming Defined Networking offers desire to change the current organization framework restrictions. By isolating the control plane and information plane SDN permits network administrator to control network conduct from a brought together single significant level control program, which is known as the regulator. Programming characterized organizing changes over the organization changes to straightforward sending components and a legitimately concentrated regulator carries out the control rationale.

The concentrated regulator has the worldwide perspective in general organization so it can choose the sending rules and introduce them on the switches.

If we are controlling organization conduct with undeniable level program, it conceivably makes the organization conduct simpler to reason about, in light of the fact that it is simpler to watch out in a solitary program and sort out the issue and discover how to control the organization. It makes it simpler to apply regular software engineering approaches which we gained from different spaces like programming dialects, programming, testing to old issues.

II. Problem Statement and Justification

- **Problem Statement** - To plan a calculation which can powerfully choose the quantity of switches that can be appointed to every regulator and allot those changes to the regulators in multi-regulator climate.
- **Problem Justification** - At whatever point a switch gets another stream, it demands the regulator to introduce fitting sending rules along the ideal stream way. A solitary regulator as a rule has a restricted asset limit and henceforth can't deal with a lot of streams beginning from all the framework switches. In customary multi-regulator climate synchronization between regulators diminishes the connection use. In multi-controller climate each dynamic regulator has at any rate one switch appointed to it.

III. Mechanisms for Handling Control Message

3.1 Collecting Flow Statistics

OpenFlow convention gives an assortment of statistics designs, including stream, port, table and line, of which stream insights is the most popular. As per the examination of DevoFlow, the exhibition of the regulator is influenced by the quantity of the stream sections and times insights are mentioned each second. In superior organization, the quantity of the stream passages is presumably enormous; for explicit applications, it is probably going to demand stream measurements for a few times each second. These activities will all burn-through the transmission capacity between the control and information planes. At the point when the regulator makes for additional estimation and investigation to the measurements, it will create further overhead on the regulator.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

As expressed previously, current arrangement is either adding control usefulness of preprocessing information to the information plane, for example, OpenSketch and DevoFlow, or conveying various leveled control sheet, for example, Kandoo. The previous requirements to alter the sending gadget, which most likely prompts an unbending execution and doesn't accord with the first aim of SDN; the last mentioned, nonetheless, at times increment the intricacy of the engineering.

In this manner, we desire to propose a natural and productive way to deal with taking care of stream measurements messages, which agrees with the standards we propose and doesn't add control functionalities of preprocessing information to the information plane. For this reason, to decrease the quantity of control messages and reduce the overhead of the regulator, we convey insights worker in Open-Flow organization, as outlined in Figure.

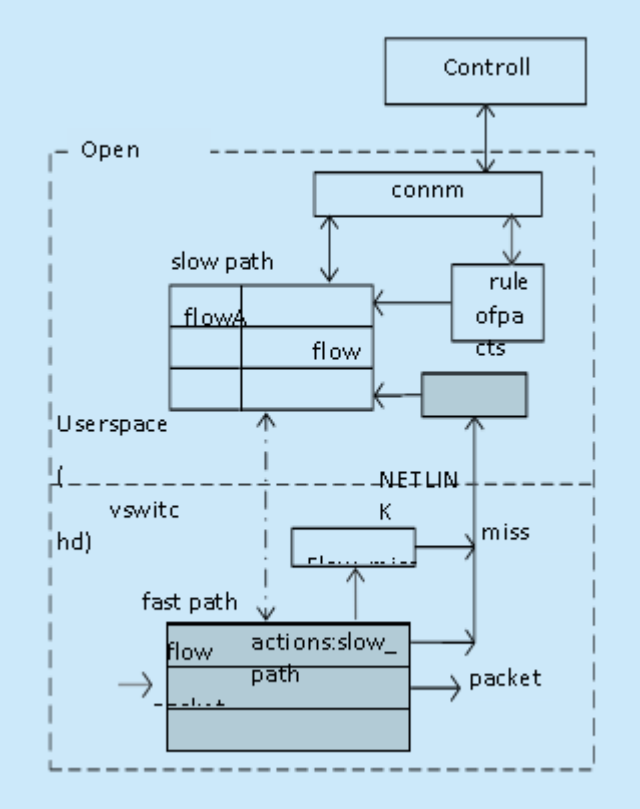


Figure 2: Basic process of Open vSwitch

At the point when the regulator needs to gather stream insights, the measurements worker follows up for the con-trawler. Also, to make the con-trawler measure continuous insights occasions, we add the usefulness of preparing stream statis-spasms to the measurements worker and use approaches which are like triggers and reports in DevoFlow to illuminate the regulator. The simplest trigger conditions are limits on the three for every stream counters in the stream table. We can likewise build more maths here to recognize traffic occasions as indicated by real prerequisite, and in our sending, we achieve the functionalities of elephant stream location and stream size dispersion in programming of the statistics worker, which are as per the possibility of SDN. In addition, we foster an application on the regulator to get the report sent from the insights worker, and settle on a choice to introduce or refresh stream passages on OpenFlow switches as per diverse requirements.

Truth to be told, all together not to alter the interior cycle or add control functionalities to the switch, the insights worker assumes responsibility for the functionalities that ought to be kept up by the OpenFlow switches or the regulator. This arrangement is identical to adding an associate handling community to the OpenFlow organization. In any case, not the same as the methodologies mentioned over, the insights worker just spotlights on gathering and breaking down measurements, and other control messages are as yet taken care of by the regulator without preparing by the intermediate control units. What we need to arrange is the division and circulation of various control streams, and OpenFlow switch itself is an elective gadget to acknowledge it. At the point when the organization scale increases and numerous regulators deal with the organization, each control space can send an insights worker close to the area regulator to reduce its overhead, to improve the adaptability in every area. In our arrangement, we foster applications for elephant stream location and stream size distribution.

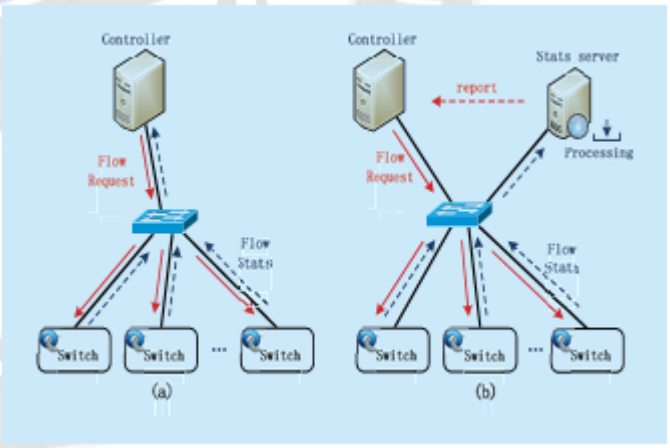


Figure 3: Adding statistics server to OpenFlow network

IV. PROPOSED ALGORITHMS

4.1 Solution to Prevent Cascading Failure

Currently we are not having any control over the assignment of new switches arriving at the SDN network and for the assignment of the switches of the initially failed controller. So here we are proposing a centralized controller by which we can control the assignment of switches. This centralized controller will only take care of the assignment of switches to the controllers in this way it will not have any traffic load. In the figure 4.1 it is shown that a single, centralized controller C

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

is having control over all the controllers, this controller is aware of the load on each controller and their capacities.

Here it is assumed that there is sufficient number of controller to handle the whole load of the SDN network. Here every time a new switch comes it requests to this central controller C for its assignment to a controller, now controller C will assign this new switch to a controller with minimum load. And before assignment, it check the load of the controller, if it is more than 80 percent of the capacity of the controller then it will not assign that switch to that controller because it may exceed the capacity of the controller. So when centralized controllers do not find any active controller capable of handling the load of the failed controller it simply makes one of the inactive controllers to active and then assign switches to that controller. In below figure controllers from 1 to 9 are active controllers and from 10 to 13 are inactive controllers.

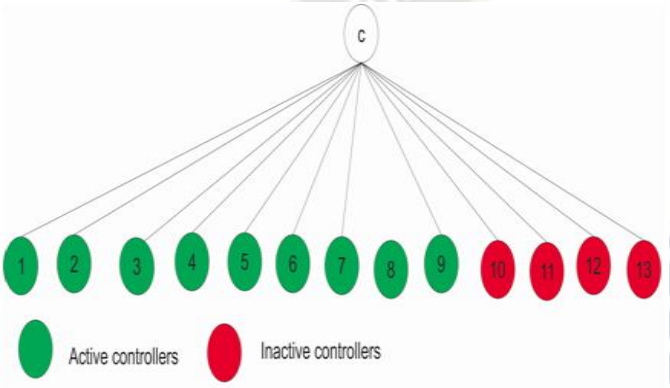


Figure 4: Multi Controller Environment with Single Centralized Controller

When a controller in this SDN network fails then every switch under that controller will be considered as a new switch and then the centralized switch starts assignment of these switches to the controllers with minimum load and if load on each active controller exceeds more than 80 percent of its load capacity then centralized controller will make an inactive controller as an active controller and then assigns these switches to that controller, and there are not sufficient controllers then centralized controller will simply discards the requests of the switches and will not assign them to any of the controller to prevent the reliability of the controller.

4.2 Assumptions

Following assumptions we have considered for this approach:

1) We have sufficient numbers of controllers, initially only one controller is active and all other are inactive.
2) Every switch is assigned to only one controller.
3) The controller can install flow rules only if it can have topological information of the whole path, i.e., all switches in that path are assigned to that controller.

4.3 Algorithms for Switch Assignment

There can be four algorithms to achieve this goal. The First is "Division algorithm", which will find that which controller should be divided. Second algorithm is "Assignment algorithm" which will decide that which switch should be assigned to which controller.

The Third is for "New switch assignment" which decides what should be done when a new switch come in the topology. And the fourth algorithm is for "Reassignment or Combining" which decides if the load decreases or switch changes their behavior dynamically, then which controllers should be stopped and how the reassignment of the switches should be done.

Each controller contains a communication matrix which maintains the count how many number of times a link between two switches have been used. Each controller also contains a load vector which contains the load imposed by each switch. Here load means the number of flow requests from that switch to the controller.

Let capacity(Ci) is the maximum threshold capacity of i th controller, and load(Ci) is the load on the i th controller.

Algorithm 1: Division Algorithm

Let T is the total number of controllers and n is the number of active controllers and m is the number of inactive controllers.

So, T=n+m, initially n=1.

```
for (i=1 to n) do
  if (load(Ci)_capacity(Ci)) then
     Make active Cn+1 and Cn+2;
     call Assignment Algorithm;
     n=n+2;
  end if
end for
```

Algorithm 2: Assignment Algorithm

```
let Pi is total number of switches assigned to a controller Ci and comm is
communication matrix or traffic matrix.
Initially max=0;
for (k=0 to Pi-1) do
   for (l=0 to Pi-1) do
      if (comm[k][l]_max) then
         max=comm[k][l];
         x=k;
         y=l;
         Assign switches Sk and Sl to same controller Cn+1;
         max=0;
         comm[x][y]=0;
      end if
  end for
end for
if (capacity(Cn+1)_ capacity(Ci)/2) then
goto A;
else
Assign all remaining switches to controller Cn+2;
end if
```

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

Algorithm 3: Algorithm for Reassignment/Combining

---

1. Every child controller share it's traffic matrix and load vector to it's parent so parent controller can have the whole topology.
2. Now load(controller)=load(left controller)+load(right controller)

if (load(controller) capacity (controller)) then
parent takes all the switches of it's child controllers under it's control and make them inactive.
end if

Algorithm 4: Algorithm for Reassignment/Combining

Every switch requests for it's assignment to root controller and root controller knows the load on all controllers, so it assigns the switch to the controller with minimum load.
Increase size of communication(traffic) matrix by 1.
Make all entries of new rows and new columns as 0.

V. RESULT AND DISCUSSION

To assess the impact of the methodologies we propose, we send an exploratory stage in the creation system of our research center.

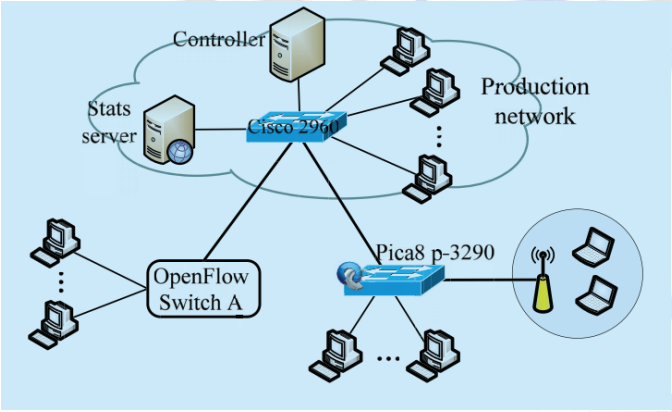


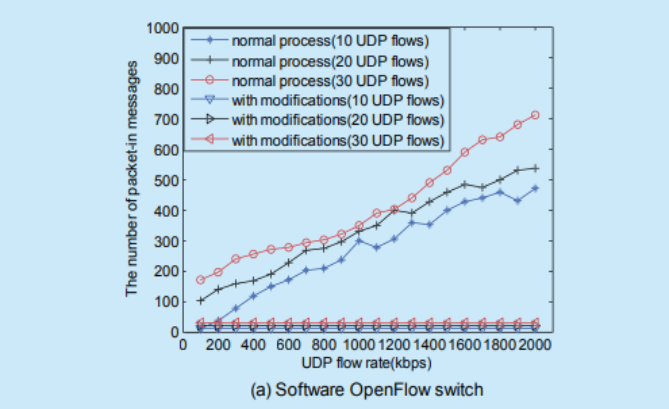
Figure 5: Experimental platform in production network

In the stage, we introduce two OpenFlow switches: one is a product OpenFlow switch (OpenFlow switch A) which introduces Open vSwitch on a host with 4 Gigabit NICs (double center 2.8 GHz processors, 2GB DDR3 RAM, Ubuntu 10.04), and the other is a Pica8 p-3290 equipment OpenFlow switch [176]. They are associated with a Cisco 2960 switch in the favorable to duction system of our research facility, and we de-ploy a regulator NOX and a measurements worker.

The regulator NOX and the insights worker are conveyed on PCs with a similar equipment and programming (double center 2.5 GHz processors, 2GB DDR3 RAM, CentOS 6.0). We include our adjustments of Open vSwitch to the product OpenFlow switch and p-3290 equipment Open-Flow switch. The adaptation of NOX and Open vSwitch are 0.9.1 and 1.9.
The systems we propose for dealing with control messages most likely influence the performance of the control and information planes. Eliminating repetitive parcel in messages is the control usefulness we include the information plane, so we measure its impact on the information plane. The reason for gathering stream insights in the insights worker is to ease the overhead of the regulator, which streamlines the dispersion of control traffic in OpenFlow arrange, so we measure its effect on the exhibition of the control plane.

5.1 The Number of Redundant Packet in Messages

We inspect the down to earth impact of dispensing with repetitive parcel in messages, and send our tests on these two OpenFlow switches individually. In the exploratory stage, we send 10, 20 and 30 UDP streams starting with one host then onto the next, and increment the stream rate from 100kbps to 2000kbps. We actualize an application on the regulator to check the quantity of the bundle in messages, and we cause it to devour a brief timeframe stretch to signify the calculation overhead. For examination, we rehash similar analyses without modifications (ordinary procedure). The aftereffects of the investigations are appeared in Figure. Clearly, both in the product switch and in p-3290, the ordinary procedure doesn't dispose of repetitive bundle in messages. With the expansion of stream rate and the quantity of UDP streams, the quantity of bundle in messages keeps up quick development. It is predictable that, if there are all the more high rate UDP streams or the handling deferral of the regulator increases, bundle in messages likely over-burden the limit of the controller right away. At the point when we add the modifications to these two OpenFlow switches, we can see that each UDP stream just creates one bundle in message.



(a) Software OpenFlow switch

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

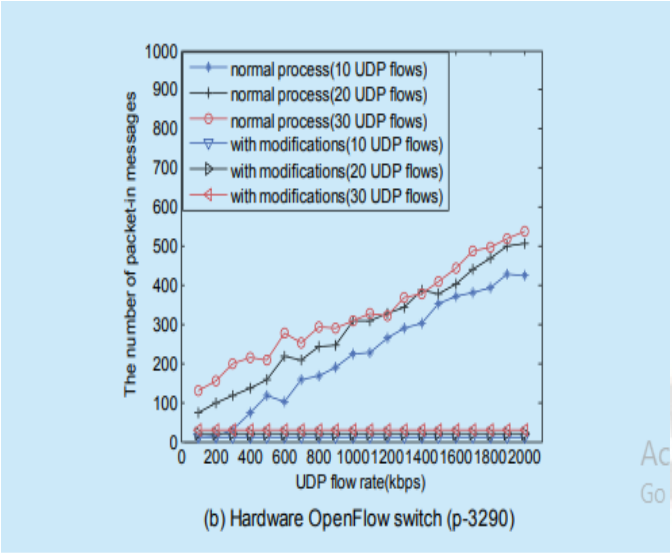


Figure 6: The number of packet-in messages in different scenarios

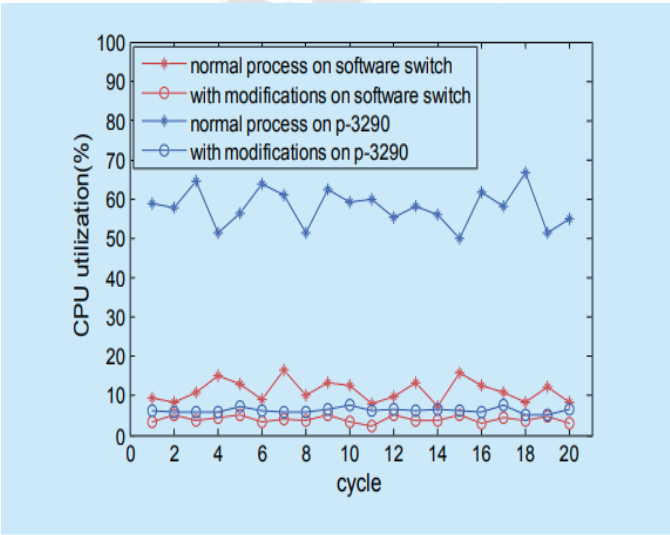


Figure 7: Comparison of CPU utilization of two switches

VI. CONCLUSION

This paper proposes and surveys large data center networks, we have thousands of switches and these switches will keep communicating with each other to share the topology and routing information which consumes bandwidth and power. To reduce this consumption, we move towards SDN. But in SDN we cannot maintain all the switches with single controller because of many reasons like a single point of failure, speed, physical limitations of a controller etc. So to solve this problem, we move towards multiple controllers. In multiple controllers switches should be assigned carefully to reduce the communication delay and to increase the bandwidth utilization. If we assign switches according to proposed solution that is according to frequent communication between switches it will reduce the communication delay and increase the bandwidth utilization. It reduces the flow set up time.

References

[1] SDN [EB/OL] [2013-9-24]. https://www.open-networking.org/sdn-resources/sdn-library / whitepapers. 2013.

[2] Yu M L, Rexford J, Freedman M J, et al. Scalable Flow-Based Networking with DIFANE[C]// ACM SIGCOMM, 2010. New Delhi, India, 2010: 351-362.

[3] Yeganeh S H and Ganjali Y. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications[C]// ACM SIGCOMM HotSDN 2012. Helsinki, Finland, 2012: 19-24.

[4] YU M L. Jose L, Miao R. Software Defined Traffic Measurement with OpenSketch[C]// Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation (NSDI), 2013. Lombard, Italy, 2013: 29-42.

[5] Schmid S, Suomela J. Exploiting Locality in Distributed SDN Control[C]// ACM SIGCOMM HotSDN 2013. Hongkong, China, 2013: 121- 126.

[6] Pfaff B, Pettit J, Koponen T, et al. Extending Networking into the Virtualization Layer. In: Proc. of the 7th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets), 2009. New York City, USA, 2009.

[7] Jose L, Yu M L, Rexford J. Online Measurement of Large Traffic Aggregates on Commodity Switches[C]// The 1st USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), 2011. Boston, USA, 2011.

[8] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: Dynamic Flow Scheduling for Data Center Networks[C]// Proceedings of the 7th USENIX conference on Networked Systems Design and Implementation (NSDI), 2010. San Jose, USA, 2011.

[9] Heller B, Seetharaman S, Mahadevan P, et al. ElasticTree: saving energy in data center networks[C]// Proceedings of the 7th USENIX conference on Networked Systems Design and Implementation (NSDI), 2010. San Jose, USA, 2011.

[10] Kotani D, Okabe Y, et al. Packet-In Message Control for Reducing CPU Load and Control Traffic in OpenFlow Switches[C]// European Workshop on Software Defined Networking, 2012.

[11] Sumit Badotra, Japinder Singh, A Review Paper on Software Defined Networking, in International Journal of Advanced Research in Computer Science, Volume 8, No. 2, March-April 2017.

[12] Pradeep Kumar Sharma, S. S. Tyagi, Improving Security through Software Defined Networking (SDN): AN SDN

based Model in International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-4, November 2019, pp 295-300

[13] Abigail O. Jefia, Segun I. Popoola and Aderemi A. Atayero, Software-Defined Networking: Current Trends, Challenges, and Future Directions in Proceedings of the International Conference on Industrial Engineering and Operations Management Washington DC, USA, September 27-29, 2018

[14] Meena, R.C.; Bhatia, S., Jhaveri, R.H.; Shukla, P.K.; Kumar, A., Varshney, N.; Malibari, A.A., Enhancing Software-Defined Networks with Intelligent Controllers to Improve First Packet Processing Period. Electronics 2023, 12, 600. https://doi.org/10.3390/electronics12030600

[15] Bari, M. F., Boutaba, R., Esteves, R., Granville, L. Z., Podlesny, M., Rabbani, M. G. Zhang, Q., and Zhani, M. F. (2013a). Data center network virtualization: A survey. Communications Surveys & Tutorials, IEEE, 15(2):909–928.

[16] Bari, M. F., Roy, A. R., Chowdhury, S. R., Zhang, Q., Zhani, M. F., Ahmed, R., and Boutaba, R. (2013b). Dynamic controller provisioning in software defined networks. In Network and Service Management (CNSM), 2013 9th International Conference on pages 18–25. IEEE.

[17] Barroso, L. A. and Ranganathan, P. (2010). Guest editors' introduction: Data center scale computing. Micro, IEEE, 30(4):6–7.

[18] Benson, T., Akella, A., and Maltz, D. A. (2010a). Network traffic characteristics of data centers in the wild. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pages 267–280. ACM.

[19] Benson, T., Anand, A., Akella, A., and Zhang, M. (2010b). Understanding data center traffic characteristics. ACM SIGCOMM Computer Communication Review, 40(1):92–99.

[20] Cervello-Pastor, C., Garcia, A. J., et al. (2014). On the controller placement for designinga distributed SDN control layer. In Networking Conference, 2014 IFIP, pages1–9. IEEE.

[21] Dhamecha, K. and Trivedi, B. (2013). Sdn issues-a survey. International Journal of Computer Applications, 73(18):30–35.

[22] Feamster, N., Rexford, J., and Zegura, E. (2013). The road to SDN. Queue, 11(12):20.

[23] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3):105–110.

[24] Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In Proceedings of the first workshop on hot topics in software defined networks, pages 7–12. ACM.

[25] Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., and Tran-Gia, P. (2013). Pareto-optimal resilient controller placement in SDN-based core networks. In Teletraffic Congress (ITC), 2013 25th International, pages 1–9. IEEE.

[26] Hoelzle, U. and Barroso, L. (2009). The datacenter as a computer. Morgan andClaypool.Hu, Y., Wang, W., Gong, X., Que, X., and Cheng, S. (2014). On reliability optimized controller placement for software-defined networks. Communications, China, 11(2):38–54.

[27] Jain, R. and Paul, S. (2013). Network virtualization and software defined networking for cloud computing: a survey. Communications Magazine, IEEE, 51(11):24–31.

[28] Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. Communications Magazine, IEEE, 51(2):114–119.

[29] Kim, H., Santos, J. R., Turner, Y., Schlansker, M., Tourrilhes, J., and Feamster, N. (2012). Coronet: Fault tolerance for software defined networks. In Network Protocols (ICNP), 2012 20th IEEE International Conference on, pages 1–2. IEEE.

[30] Kirkpatrick, K. (2013). Software-defined networking. Communications of the ACM,56 (9):16–19.

[31] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., et al. (2010). Onix: A distributed control platform for large-scale production networks. In OSDI, volume 10, pages 1–6.

[32] Kreutz, D., Ramos, F., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. In Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking, pages 55–60. ACM.

[33] Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey Proceedings of the IEEE, 103(1):14–76.

[34] Limoncelli, T. A. (2012). OpenFlow: a radical new idea in networking. Queue,10(6):40 Phemius, K., Bouet, M., and Leguay, J. (2014). Disco: Distributed multi-domain SDN controllers. In Network Operations and Management Symposium (NOMS), 2014IEEE, pages 1–4. IEEE.