International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Analysis of Vertical Scalability for Controller Placement in Software-Defined Networking and its Implementation using Tree based Architecture

Ritesh Jain*[1], Dr. Pradnya Ashish Vikhar[2]

Submitted: 09/01/2024 **Revised**: 15/02/2024 **Accepted**: 23/02/2024

Abstract: Software defined networking decouples data plane and control plane. Controller, which is a centralized control plane, is responsible for performing the control operations. By doing this it simplifies network management and provides the opportunity for fast innovation and development. By separating data plane and control plane each can evolve independently. It facilitates the introduction of new services much more easily. Control plane works as a network's brain. It allows us to program the network in our own way. Every switch should be connected to one controller. For big data centers, there is needs of multiple controllers. By proper controller placement the reliability of the data center network can be increased. The main motivation is to minimize the communication delay. For large data centers, we require multiple controllers, when we use multiple controllers, we need to decide which switch should be assigned to which controller and how many switches should be assigned to a particular controller? In big data centers, switches change their behavior dynamically. So, the switches also should be assigned dynamically to the controllers. Hence the aim is to design an algorithm which can assign switches dynamically to the available controllers and can improve the overall performance. In this work a solution to switch assignment problem and cascading failure of controllers in multi-controller environment is given. A hierarchy for multiple controllers is proposed which ensures that cascading failure of multiple controllers is not possible. To assign the switches to multiple controllers frequently used links are taken as parameter, it reduces the flow set up time and load on the links between controllers. A comparison between random switch allocation and switch allocation according to proposed solution is done which shows flow set up time is very less in proposed solution.

Keywords: *Controller Placement, Software Defined Networking, Tree based Architecture*

1. Introduction

Present computer organizations are using huge and complex networks, there are many equipments involved in computer networks such as firewalls, routers, switches, network address translators, intrusion detection systems. In current scenario we cannot store all our information on the local systems and due to the increasing volume of the data companies have moved towards a new concept which is known as a data center.

Data center reliability in the real world is strongly depends on the organization running the data center, not just on the design. The heat generated by all equipment is removed by datacenter cooling systems. There must be some hierarchy of loop system in a cooling system for removing the heat, each time the hierarchy brings a cool medium that warms up from some heat exchange and again cooled back somehow.

1.1 Software Defined Networking (SDN):

The control plane refers to the collection of functions inside a network that dictate its behaviour. Typically, it is represented by a single, advanced software controller that serves as the central intelligence of the network, coordinating its operations. In contrast, the data plane encompasses the network services that are specifically designed for the purpose of forwarding traffic, effectively managing the physical conveyance of data. The SDN controller communicates with the network switches below it to enable the control and data planes to interact. This ensures that the control plane's strategic decisions are effectively implemented by the data plane in the physical flow of network traffic.

1.2 Tools and Platform:

When Floodlight is executed, the activities of both the northbound and southbound APIs from the controller are activated. Any application has the capability to communicate with the controller by sending an HTTP REST command. On the other hand, outside the network, the Floodlight module will begin listening on the designated TCP port given by OpenFlow for connections from the OpenFlow switches. At now, Floodlight is also compatible with the OpenFlow 1.0 protocol.

Floodlight is an SDN controller that is both user-friendly and powerful, thanks to its flexible Java development environment and enterprise-grade core engine. Mininet uses Python as its underlying programming language. Mininet

[12] *Department of Computer Science and Engineering*
[1] *Research Scholar, Dr. A. P. J. Abdul Kalam University, Indore*
[2] *Research Supervisor, Dr. A. P. J. Abdul Kalam University, Indore*
E-mail Id: [1]rit.rit1@gmail.com, [2]pradnyav123@gmail.com
** Corresponding Author: Ritesh Jain*
Email: rit.rit1@gmail.com

operates with an authentic kernel and switch, enabling the execution of genuine software and application code on a single system. OpenFlow encompasses several pre-existing capabilities, rendering it advantageous for the development, implementation, and dissemination of diverse elements inside the Mininet programming environment.

The segregation of the data plane and control plane enables network operators to manage network behaviour using a centralised, singular, high-level control programme. The present architecture consists of routers, a control plane, and numerous functionalities. The Management plane employs many protocols such as Simple Network Management Protocol (SNMP), Telnet, Hypertext Transfer Protocol (HTTP), Secure HTTP (HTTPS), and Secure Shell (SSH). Routers use these commands to determine the network topology and determine the actions of physical and virtual switches. The behaviour of these switches is contingent upon the requests made by applications via the northbound APIs. The first advantages of SDN will mostly arise from the implementation of network virtualization, enabling more flexibility in network segmentation and utilisation.

The switches forward packets as instructed by the controller. Software Defined Networking has the potential to overcome the existing limits of network infrastructure. SDN enables network operators to manage network behaviour using a centralised control programme, referred to as the controller, by separating the control plane and data plane. Software defined networking transforms network switches into basic forwarding devices, while a logically centralised controller executes the control logic.

The centralised controller has a comprehensive perspective of the whole network, enabling it to determine the forwarding rules and implement them on the switches.

By using a high-level programme to govern network behaviour, it has the ability to enhance the comprehensibility of network operations. This is due to the convenience of examining a single programme to identify issues and determine the means to regulate the network. It makes it easy to use traditional computer science methodologies which we learnt from diverse disciplines such as software engineering, programming languages, testing to old issues.

2. Problem Statement and Justification

Analysis of Vertical Scalability

The controller may experience overload as a result of the excessive number of control messages originating from the data plane. This issue is especially troublesome in smaller organisations, when the controller's performance is constrained by the hardware and control applications it executes. This bottleneck arises particularly when certain hosts generate a substantial amount of control messages within short time periods, posing a danger of overwhelming the controller. The main forms of control communications that may greatly increase the operating burden of the controller are packet-in messages and flow statistics messages.

In order to reduce the workload on the controller, a common approach is to integrate control operations directly into the data plane or assign them to local controllers. This method enables more effective and localised handling of packet-in messages, hence reducing the need for substantial controller involvement. When it comes to flow statistics messages, it is more efficient to consolidate this data inside the data plane by using techniques such as sampling, triggers, and summaries.

An analysis of the consequences of incorporating control functions into the data plane or local controllers uncovers several approaches for improving vertical scalability. To achieve vertical scalability, the most direct method is to include control capabilities directly into the data plane. This entails creating a two-layered system for handling control signals, as shown in figure 1. This approach not only simplifies the management of control messages but also enables the use of more scalable network designs.

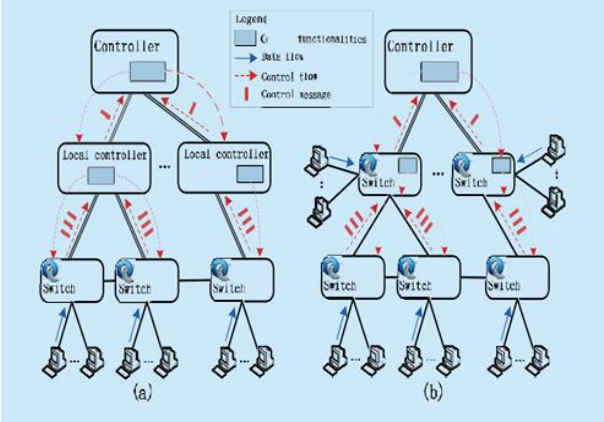

Fig 1. Two-layer SDN architecture

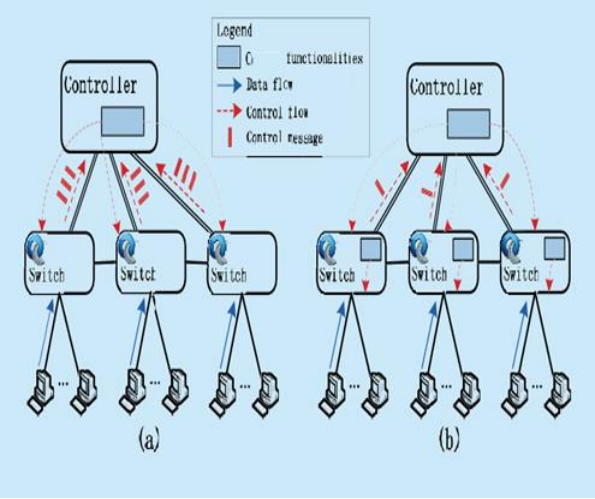

Fig 2. Three-layer SDN architecture

Figure 1 presents the fundamental concepts of Software-Defined Networking (SDN), demonstrating a centralised control paradigm in which the controller oversees all control operations and directs the OpenFlow switches in data transfer according to its instructions. Figure 2 examines a novel method for handling individual data packets using localised tactics such as duplicating rules or performing direct actions inside the switch. This approach also allows for the consolidation and compression of flow data at the switch level. This approach avoids the need of a dual-layered design by ensuring that the maximum delay in processing control signals matches the delay shown in Figure 1. Practical implementations of this idea include DevoFlow and OpenSketch.

The discussion on vertical scalability in SDN revolves on two primary approaches that include including intermediate control units into the network's design. Figure 2 depicts a configuration in which local controllers, situated between the main controller and the switches, manage proximal control messages. These local controllers may directly conduct tasks that do not need global supervision, such as real-time data collecting, with this setup. The central controller maintains its responsibility for supervising global operations such as geographic exploration and algorithmic modifications. Kandoo, on the other hand, serves as a realistic demonstration of this architecture, but lacks the inclusion of additional local controllers. Specific switches provide control functions that allow them to oversee neighbouring switches, therefore incorporating control elements into the data plane to optimise traffic management. These specialised switches have a greater control duty compared to their ordinary counterparts, as shown by DIFANE, which delegates control chores to them.

Examining these control plane models demonstrates that including control features inside the data plane and using local controllers may decrease the burden on the central controller and improve network adaptability. Nevertheless, this integration amplifies the operational intricacy inside the data plane and can need an OpenFlow extension to directly configure these control functions from the controller, deviating from the standard protocol established by the ONF. This method of interacting with neighbouring controllers entails preserving data and control plane links, guaranteeing a uniform network state across several control tiers. Although the traditional OpenFlow interface has its limits, the advancement towards an OpenFlow-based SDN framework highlights the significance of protecting the control plane's integrity and assuring effective processing of data plane packets. In order to facilitate the capacity to scale upwards, three concepts are suggested:

Control functions need to be located in the control plane, unless their incorporation into the data plane may improve processing efficiency without sacrificing the capabilities of the hardware and software.
Control function integration must preserve the fundamental processes of the data plane, ensuring that network devices remain simple and efficient inside an SDN environment.
The inclusion of data analysis capabilities in the data plane should not have a negative impact on the correctness or validity of operations, nor should it substantially increase the burden of the control plane.
The separation of control and data planes is crucial as SDN continues to develop, particularly with OpenFlow as its foundation. However, the progress involves integrating certain control functions into the data plane, such as the replication of rules seen in DevoFlow. Open vSwitch, a virtual switch that is compatible with OpenFlow, employs specialised treatment of stream entries to enable fast packet processing. This feature, similar to duplicating rules, is well-suited for the software-defined architecture, improving data transfer rate without overloading the controller. Therefore, these concepts play a crucial role in directing the SDN's transformational path towards a more scalable and efficient network administration.

3. Mechanisms for Handling Control Messages

In order to tackle the issue of handling excess packets in messages, especially in the context of connectionless protocols such as UDP, it is necessary to devise a method to avoid overburdening controllers due to high-rate UDP streams or intentional activities by malevolent entities. To address this problem, it is necessary to enhance the data plane's capacity to differentiate between various streams. Nevertheless, the OpenFlow protocol does not explicitly delineate the processes for attaining such difference.

To properly address this problem, it is crucial to provide either the data plane or local controllers more control capabilities in order to efficiently handle different streams. As per the first recommendation, the responsibility of recognising and controlling various data flows should be handled by the data plane, in keeping with the operational principles of the OpenFlow protocol. This indicates that the data plane must have the capability to efficiently separate and regulate different streams of packets when it receives instructions from the controller.

Upon examination of Open vSwitch (version 1.9), it is evident that a hash map is used for stream identification. This hash map organises streams into buckets, allowing for the efficient separation and management of individual streams. By implementing targeted alterations to the switch's software, it becomes feasible to eliminate superfluous packets from messages, harmonising with the current programming framework of the architecture without modifying the fundamental processes of data transmission. Although these modifications resolve the problem of excess packets, it is essential to guarantee that they do not violate the criteria of preserving simplicity and effectiveness in

network device operations inside an SDN environment.

Moreover, the advancement of SDN architecture signifies a rise in the utilisation of white-box switches that may implement OpenFlow functions at the hardware level. These switches, known for their open software ecosystem, generally consisting of Linux and Open vSwitch, represent a progressive direction in the development of Software-Defined Networking (SDN). Modifications made to Open vSwitch may be immediately transferred to various hardware switches, hence expanding the range of our solutions.

Open vSwitch consists of two main components: a user space component that connects with the controller, and a kernel module that manages the receipt and routing of packets. The system has two distinct flow tables to optimise packet processing: a high-speed channel for instant processing and a low-speed path for more intricate processes. By improving the administration of the stream table in the user space and allocating distinct IDs to each stream, the switch can better handle and distinguish across streams. This advanced technique attempts to enhance packet handling without compromising the integrity of the data plane's operations, while adhering to the principles of successful SDN management.

4. Algorithms

Solution for Dynamic Switch Assignments:

Tree Based Architecture As a solution a tree-based scheme is proposed in which there will be a hierarchy between controllers, all the switches will be connected to bottom most controllers. Every switch must be connected to at least one controller. Every controller has two child controllers and every child controller will share their topology information to its parent controller. Every time a switch receives a new flow request it will ask its controller to install flow rule, the controller will now check the flow rule from its topology information. If it finds it will install the flow rule otherwise it will forward the flow request to its parent.

Now the same procedure will repeat at the upper layer controller and this will continue until the rule is installed or the request reaches to the root level controller, since the root level controller has the global view of the topology it will definitely install the required flow rule, if the requested path is available otherwise the rule will not be installed. Here we are considering two types of controllers, active controller is the controller which is assigned at least one switch, otherwise the controller is known as inactive controller. Inactive controllers always keep listening for new switch assignment, it saves power and resources.

If we use single, centralized controller for controlling whole network then it suffers from a single point of failure. If centralized controller fails, then whole network will fail (collapse) but in tree-based architecture if one controller fails, then also the network will work fine in two parts. As shown in figure 3.

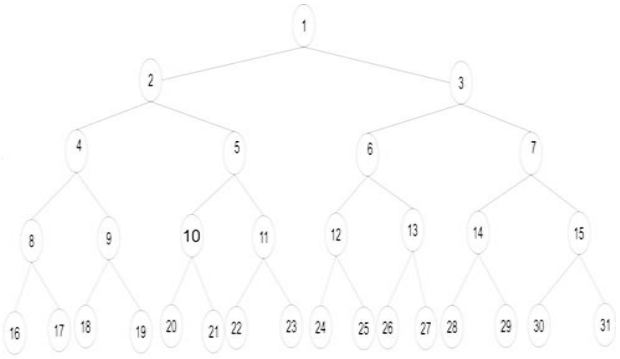

Fig 3. Tree topology of controllers

In the hierarchy of controllers shown in figure 3 if controller 1 fails as shown in figure 4, then also the network will work in two halves that is controller 2 and controller 3 will become root now and flow rules for the paths under the controllers 2 and 3 will be installed properly.

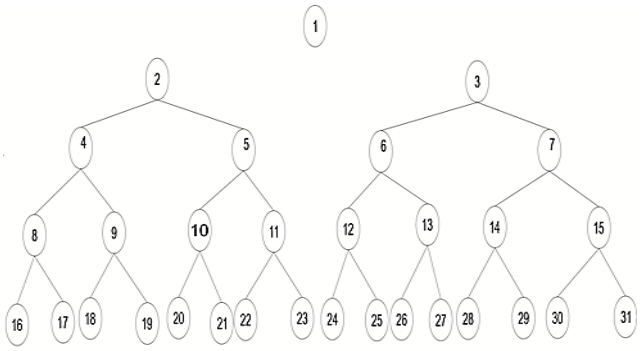

Fig 4. Tree topology when controller 1 fails

In current architecture failure detection is also a big problem, if a controller fails, then there is no exact method which can find the failed controller in less time, and after finding the failed controller it is also a tough task to assign switches under that controller to other controllers, here the controller can get overloaded because the switches are assigned randomly. But in proposed architecture each controller sends echo messages to its parent controller and receives echo messages from its child controllers. So maximum 3 echo messages will be required to find the failure of any controller, and since it is an organized structure so the switches under the failed controller will be easily equally distributed among other active controllers by the root controller. Here the switches are assigned based on equal load so there is no chance of controller failure on the exceeding of load.

Initially, it seems that we are using extra controllers in tree topology, but these are necessary because we need a proper mechanism through which we can dynamically assign switches when a controller comes to its full load, on

condition of the divide we need hierarchy and at the time of combining switches i.e., when load decreases, we need to reduce the number of active controllers.

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) All controllers are connected in tree topology. Every controller has two child controllers, except the leaf level controllers.

Algorithm for Tree Based Architecture

1) Initially there will be only one controller, every switch will request to this controller for assignment.

2) When the controller reaches to its maximum load it activates its two child controllers and divide its switches among both child controllers, in such a way that load on both the child controllers will be nearly equal.

3) Both the child controllers frequently share their load to their parent controller, here the parent controller knows about total load on the network under their control.

4) Now the parent controller can decide on the basis of total load that whether the child controllers should remain active or it should take all the switches under its control and make them inactive.

5) Every new switch request to route controller for its assignment, since the root controller knows load on every controller so it will assign the switch to the controller with minimum load, in this way the load will be divided among all the controllers equally.

5. Result and Analysis

Mininet is used to create topologies for data center networks. For this purpose, Fat-Tree topologies and Random topologies are used. In Fat-Tree topology, there are three layers of switches. Core layer is the uppermost layer, and the edge layer is the lowest layer, the middle layer is the aggregation layer. The hosts or application servers are connected to the switches of edge layer. In Random topology switches are connected randomly to each other.

Since none of the switch knows the route information it always forwards the initial route request to the controller, and then controller installs the flow rules. In this work Floodlight is used as SDN controller, which will compute and installs rules in switches based on incoming flow requests from switches. On the controller we can measure the load that is the number of flow requests coming in a specific time unit (Here 1 second is considered).

Following image shows the some of the topologies used:

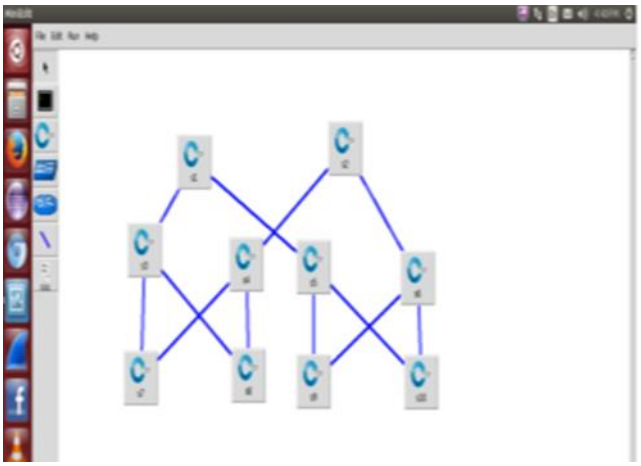

Fig 5. Topologies -1

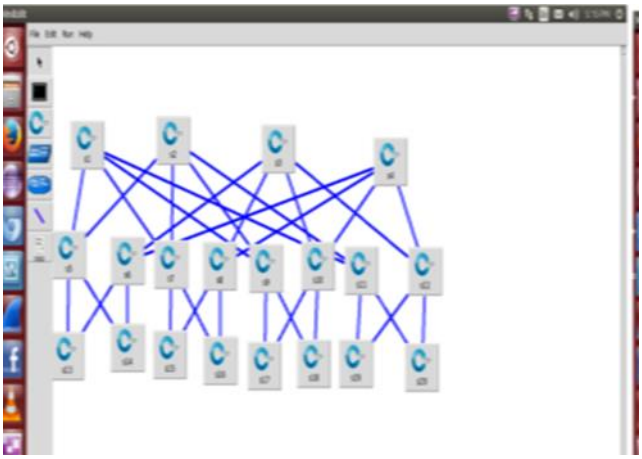

Fig 6. Topologies -2

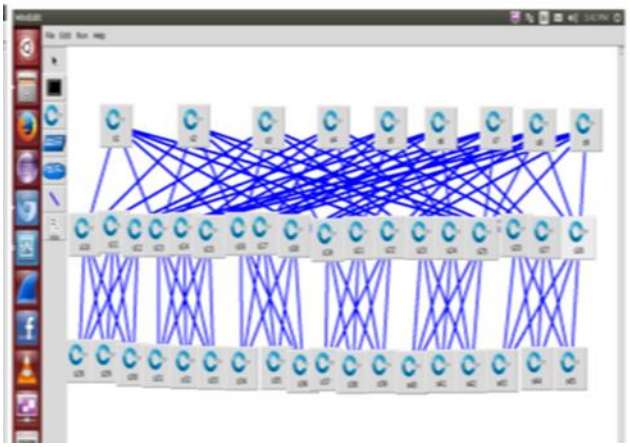

Fig 6. Topologies -3

The current method of multi controller placement is suitable for static switch assignment and proposed solution is suitable for dynamic switch assignment, as it's an organized structure so easy to control. The Proposed scheme is suitable for data center networks as there is no need of consideration of length in data center networks, and data center networks are small in area so maintaining them is an easy task. Let every controller have the capacity of handling traffic from 100 switches at a particular time T and let say there are 1000 switches.

The following graph shows the number of controllers required at any point of time in the network depending upon the number of flow set up requests.

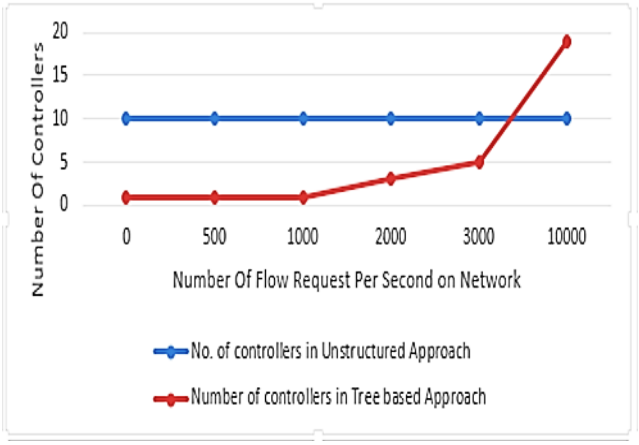

Fig 7. Number of flow request per second on network

If the initial average number of requests per second is 0 or 500 or 1000 requests and then traffic varies dynamically then in non-organized way, which is being used currently we will require minimum 10 controllers. In tree topology we will require only one active controller.

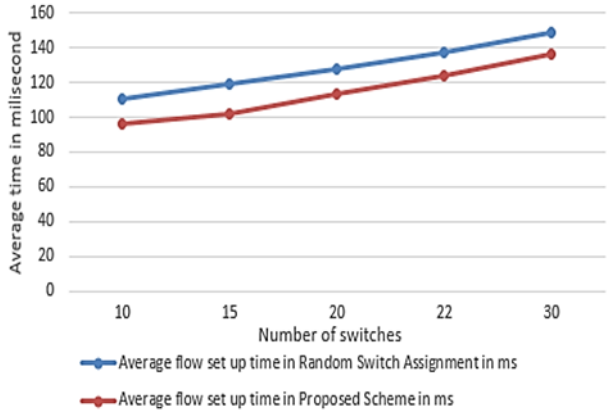

Fig 8. Assign switches randomly to any controller

From the above graphs it is clear that if we assign switches randomly to any controller then the flow set up time will be greater as compared to proposed approach of assigning the switches which are connected to the most frequently used link. The reason behind it is that in random assignment of switches to controller it is possible that to fulfill a flow set up request more than one level of tree hierarchy need to traverse the request, But if we assign the switches according to proposed approach, then maximum the flow set up requests are satisfied at bottom most layer of the hierarchy, in this way the average flow set up time reduces.

6. Conclusion

Within the realm of Software-Defined Networking (SDN) architecture, there is a growing inclination towards implementing a distributed control plane in order to improve scalability. Nevertheless, existing approaches to enhance scalability often include integrating control features into the data plane, hence potentially complicating its design. Given the ongoing development of SDN, we propose three fundamental concepts for incorporating control capabilities into the data plane. In accordance with these principles, we have devised two techniques with the objective of reducing control messages in OpenFlow-based SDN settings. The experimental findings confirm that our suggested methods successfully reduce the burden on the controller and improve the scalability of the network.

Author contributions

Ritesh Jain: Conceptualization, Methodology, Software, Field study, Data curation, Writing-Original draft preparation, Validation **Dr. Pradnya Ashish Vikhar:** Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

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.