A Mathematical Model for SDN Control Plane Scalability Evaluation Based on Controller Utilization

Firas Zobary*[1], Li ChunLin**

** School of Computer Science and Artificial Intelligence, Wuhan University of Technology*
Wuhan, Hubei, China (e-mail: firas_zobary@hotmail.com)
*** School of Computer Science and Artificial Intelligence, Wuhan University of Technology*
Wuhan, Hubei, China (e-mail: chunlin74@aliyun.com)
[1]Corresponding Author: Firas Zobary

Abstract: The increasing number of users on internet, clouds, and data centers should be followed by more efforts to change the behavior of traditional networks. Software-defined networks give more flexibility to the network, but the scalability of SDN is still a problem for network designers and administrators about which architecture is suitable to be applied in their network. As more studies study the scalability problem, controller utilization as a scalability metric for different SDN control planes still doesn't have a deep analysis which is the novelty of this work. In this paper, we propose a mathematical model for SDN control plane scalability evaluation based on controller utilization as a metric. This paper employs mathematical techniques to examine and contrast various architectures. The numerical results conclude that the decentralized global view has the worst utilization, while the hierarchical root controller has the best, and the local decentralized control plan has almost the same utilization as the hierarchal leaf controller. These findings have significant implications for network designers and administrators as they suggest that implementing a hierarchical control plane can optimize the network scalability, ensuring efficient network management and resource utilization. This paper not only presents a mathematical model for evaluating SDN scalability, but also provides valuable insights into the implications of the findings. Network professionals can leverage this knowledge to make informed decisions and design more efficient and scalable SDN. The validity of the findings presented in this study is further substantiated through numerical assessments and achieved by proposing a novel method.

Keywords: Control Plane, Controller, Scalability, SDN, Utilization.

1. INTRODUCTION

With the increasing development of networks, internet, and cloud services, traditional networks start to show their limitations when they move to new concepts like 5G networks, mobility, server virtualization, or IoT. Because of the huge number of users as well as their needs related to the quality-of-service speed, researchers have started to rethink today's network architecture. The traditional architecture we know nowadays suffers from complexity and difficulties when it comes to configuring, device installation, and vendor compatibilities. Adding or removing a switch or a router in a network is a wasted time job as the administrator should deal with hundreds and maybe thousands of network devices (Sezer et al., 2013). Software Defined Network (SDN) is a new paradigm of modern network architectures that creates an operative relation and communication between network devices, clouds, mobile phones, internet of things, and data centers (Yurekten and Demirci, 2021). In SDN, the control layer is decoupled from the physical layer, and the network device is abstracted from the upper layer, which is called "controller." This abstraction brings many advantages to the new architecture as it starts to be more flexible, and programmable, and gives the ability to centralized network monitoring. Three layers compose an SDN architecture: the application, control plane, and data plane. The main part of the SDN is the control plane (controller) which has the core responsibility of managing the data plane and flow control. In the data plane, network devices such as switches and routers are responsible for executing the controller instructions about packet forwarding and data flows from the source to their destination. As a new paradigm, SDN are facing multiple challenges that should be addressed during its implementation. Scalability is one of the most important challenges that should be focused on to ensure the effectiveness of processing the increasing demands on network resources. Decoupling data and control planes may need more efforts to secure the controller from unauthorized access to ensure the privacy of sensitive data. Moreover, SDN deployments often involve a mix of vendors and protocols, so the interoperability can be challenging and therefore, it's essential to establish standardization protocols to facilitate the deployment of heterogeneous environments. Also, the transition from traditional network to SDN can affect the recent network infrastructure and add more costs to deploy new hardware, software and training. Addressing the previous challenges requires ongoing researches to ensure a successful adoption of SDN. Thus, most researchers and studies are focusing on controller-related issues as the controller is the key part of SDN. These issues vary from choosing the suitable controller for the network to evaluating the controller's performance under many conditions. Control plane scalability is one of the most important issues that researchers should focus on. In general, three classifications of control plane are introduced:

centralized, decentralized, and hierarchal design (Schmid and Suomela, 2013). Only one controller is implemented in centralized architecture, and it's responsible for all the data flow management and has a global view of the whole network. However, this design is appropriate for some small-sized networks, but as the incoming flows increase and the number of network devices grows, the controller will become a bottleneck and a single point of failure as, at some point, it will not be able to handle all the requests (Benson et al., 2010). In decentralized architecture, some multi-controllers are distributed in a flat design. The hierarchal architecture also uses multi-controllers but in tree distribution. That is, the network is divided into several clusters, and each cluster is controlled by only one leaf controller, which has a local view of its network. The leaf controllers at the end are managed by an upper-layer root controller that has a global view of the whole network. Our study aims to mathematically model and evaluate the control plane utilization in these three basic SDN architectures to further address their scalability behavior. The mathematical model for SDN controller utilization, while primarily designed to assess scalability, can indeed be used to analyze the task execution time including calculation time, I/O time and transmission time and therefor impact service response time. Interpreting the controller utilization data is important to understand how the workload on SDN controllers can impact the time it takes to perform computational tasks, handle I/O operations, and transmit data across the network. As the controller utilization increases, the time taken to perform calculations may also increase, and that may indicate a resource contention (CPU, memory) between different tasks, so the high utilization controller needs to be allocated by more resources to process calculations faster. Also, the workload on a controller can directly affect tasks involving input/output operations such as reading or writing data to storage devices. Moreover, controller utilization can directly affect the transmission time and network latency. As the high controller utilization might lead to network congestion due to an increased number of control messages being transmitted. This congestion can significantly impact the time taken for messages to travel across the network, so it's very important for network administrators to analyze how congestion during peak utilization affects transmission time and, consequently, task execution time. This evaluation is necessary for network operators and designers before building their network topology and choosing the most suitable network architecture. Controller performance evaluation has been studied in many researches. A comparison between SDN and non-controller networks was made in(Alraawi & Adam, n.d.) in terms of throughput and delay, while seven controllers are evaluated against several performance metrics (Lunagariya and Goswami, 2021). Controller performance in special network designs such as wireless and IoT are evaluated in scholars (S. Islam et al., 2019; Urrea and Benítez, 2021; Zhou et al., 2022). The main goal of the above studies is to help the network administrator to choose the most suitable controller for the network. Before controller deployment, the operator must know in advance the controller's ability to be scaled and can handle the increasing number of flows and network devices. In this scenario, the distributed controller is much better than the centralized one as this architecture provides high throughput and low delay without being a bottleneck (Othman et al., 2017).

The novelty and contributions of this study are as follows:

1. The proposed mathematical model for evaluating SDN control plane scalability is based on controller utilization as a metric.
2. The use of mathematical techniques to examine and contrast various SDN control plane architectures.
3. The conclusion is that the hierarchical root controller has the best utilization, while the decentralized global view has the worst utilization, and the local decentralized control plan has almost the same utilization as the hierarchal leaf controller.
4. The numerical assessments further substantiate the validity of the findings presented in this study.

2. RELATED WORKS

Scalability measurement is not a new idea. Several works have been done to measure the scalability of different systems, and most of them focused on scalability measurement in distributed systems and algorithms as well, but scalability evaluation in terms of control plane utilization is still widely unexplored. An experiment was conducted on the popular Ryu controller, a Python-based controller, to measure scalability metrics (Asadollahi et al., 2018). The experiment involved a test bed with 6 switches arranged in a mesh topology, and the host was able to support up to 300 nodes. The experiment measured the throughput performance, and it was found to be stable. However, when compared to scenarios with 100 and 200 nodes, the results were negative, indicating that the performance degraded as the number of nodes increased. These findings suggest that implementing the Python-based controller over Ryu may not be optimal for a larger network.

A study was conducted in (Satre et al., 2021) to analyze the performance of firewall applications, using a custom topology and firewall scripts written with the aid of another firewall script. The objective was to filter traffic based on their parsed headers, and TCP/UDP traffic was tested to be blocked using POX as a reference. Round trip time (RTT) was evaluated, with no significant variation observed when the firewall switch application was not used. However, when testing throughput with and without the firewall application, variation in results was noted. Latency variation and jitter time were also found to be higher when compared to the default switch application. The purpose of this experiment was to examine the performance impact of running a switch application with an embedded firewall script to filter traffic based on rules.

The purpose of the study (M. T. Islam et al., 2020) is to assess the performance of the Ryu controller in terms of bandwidth, throughput, round trip time (RTT), jitter, and packet loss. To evaluate the Ryu controller's bandwidth, iPerf3 tool is used to generate TCP traffic and analyze the bandwidth between three hosts arranged in a single topology. Throughput was measured between nodes acting as client and server, and a graph was created to depict the highest and lowest values of TCP traffic. The minimum, maximum, and average RTT times between hosts were also calculated. The study found no fluctuations in jitter results, which is important for maintaining connection

reliability, as high jitter can disrupt the connection. Additionally, the packet loss did not exceed 1%, as exceeding this threshold can lead to TCP re-transmission, which negatively impacts bandwidth.

A performance evaluation of five controllers, namely Libfluid, ONOS, OpenDaylight, POX, and Ryu, was carried out in (Abdullah et al., 2018) to assess their end-to-end delay and throughput. The evaluation was conducted using a linear topology with a varying number of switches, ranging from 8 to 64. In this topology, one host was designated as a server, and the other as a client to test throughput. It is worth noting that no protocol can ensure 100% throughput to bandwidth, and delay, generally referred to as round-trip time (RTT), starts to increase as the number of switches load increases. Among the five controllers tested, ONOS had the lowest delay value, while Libfluid had the highest delay value. Libfluid and POX exhibited the highest throughput but stopped responding at 1024 hosts, whereas the other controllers stopped at 512. The performance analysis concluded that the throughput and delay increased with an increased number of switches in the linear topology.

Experimental comparison and evaluation in (Badotra and Panda, 2020) were conducted on Mininet using four different topologies, including single, linear, tree, and custom with varying numbers of hosts, ranging from 10 to 1000, to assess the performance of seven different controllers, namely Terna, Floodlight, POX, Ryu, OpenDaylight, and ONOS. The primary objective was to determine the controller's performance by analyzing whether it took less time with an increased number of nodes, which is generally considered indicative of good performance. OpenDaylight outperformed the other controllers in terms of bandwidth transmission and jitter delay. The analysis showed that OpenDaylight took less time with minimum RTT analysis for both 10 hosts and 1000 hosts in a single topology and even in a custom topology, indicating that OpenDaylight performs better based on the minimum time analysis.

Because of the increasing number of users and network devices, the scalability of the control plane is a very important topic that should be studied by literature and researchers. RYU controller scalability is addressed by implementing different topologies (Cabarkapa and Rancic, 2022). It is concluded that distributed controllers offer high throughput and the future of SDN relies on distributed control plans. The proposal (Yeganeh et al., 2013) was to design a control plane with more than one controller and make these controllers communicate and cooperate to process incoming and outgoing data flows. This approach doesn't need a special network device. Hyperflow (Tootoonchian and Ganjali, 2010), Kandoo (Hassas Yeganeh and Ganjali, 2012), and Onix (Koponen et al., 2010) are three major distributed controllers, each with a different distribution strategy. Hyperflow uses a decentralized controller architecture with only one layer containing all the controllers that share the same view of the whole network topology, and they know everything about all the events that happened in that network. This design has the advantage of decreasing the processing time for the controller and increasing the throughput, but its disadvantage is resource consuming as the controller may process and store unnecessary information. Onix architecture consists of many clusters on which multiple Onix instances are running. The network state is distributed among these multiple instances. An Onix instance is responsible for distributing the network state to other instances within the same cluster. In Onix, each controller with its managed network is introduced as one logical node. The third controller distribution architecture is the hierarchal architecture, and Kandoo is the major controller that is adopting this design. In Kandoo, the multiple controllers are distributed in two layers, leaf controllers and root controllers. Leaf controllers communicate directly with network devices and know only about their network and nothing about network-wide information, whereas root controller has a global view of the network and can manage the local controllers.

The works mentioned above have a general view of SDN scalability and were only studied in experiments, not as mathematical modeling. Furthermore, all of the previous studies discussed the performance evaluation of a limited number of SDN controllers and didn't mention the control plane itself, while in our work, we mathematically evaluate the controller utilization as a scalability metric in different control plane architectures.

3. PROBLEM MODELING

3.1 SDN Controller Scalability

We can consider SDN as a distributed system, and most distributed systems should be scalable. They should be deployable in a wide range of scales, and scalability here means not just to operate well but also to operate efficiently over any range of configurations (Jogalekar and Woodside, 2000). One of the important scalability metrics is the controller utilization, the probability that the controller is busy, and the ability of the controller to handle the increasing number of incoming flows from network devices and process it efficiently. We use distributing system utilization as the controller utilization (U) which is defined as:

$$U = \frac{\lambda(h)}{\mu(h)} \tag{1}$$

where (h) is the number of hosts, $\lambda(h)$ is the mean arrival rate of the controller, and $\mu(h)$ is the mean processing rate of the controller. As λ and μ have the same unit of flows per unit time (e.g., flows per second), therefore, the controller utility is a dimensionless value because it presents the utilization as a ratio or percentage indicating the efficiency or utilization level of the controller in handling incoming flows.

Since the flow initiation request arriving at the controller follows a Poisson distribution with the average λ (Ross, 2014), we can consider each SDN controller as a queue with a notation of M/M/1 as both the interarrival time and service time are exponentially distributed. In distributed systems, if $U \geq 1$, the queue becomes unstable, the wait time in the queue will be infinite, and the system needs additional servers to get back to stability. In SDN, different flows can pass the network from the network devices to the controller and vice versa, such as network view, packet-In messages, and network failure messages. The most important flow is the initiation flow request shown in Fig. 1. When the network device receives a

new flow with a new source or new destination, it is unknown how to deal with this flow because there is no flow entry matching in the flow tables, so it sends a flow initiation request to the controller which process that flow and installs it on the flow table and broadcast that flow entry to the network devices under its control simultaneously. In this case, each flow can generate one flow initiation request to the controller. For this reason, we will focus on controller utilization during the initiation requests processing level.

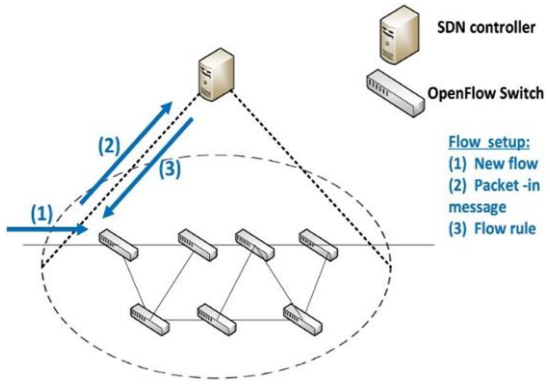


Fig. 1. Flow Initiation Request.

3.2 Controller Computation Process

When the controller receives a flow request from a switch, it processes that request to compute a network path for it and sends that new path as a flow entry to the corresponding network devices. Three parameters are involved in the path computing process; controller CPU power (P), the routing algorithm used by the controller, and the number of network nodes (V). The CPU power reflects how many operations the controller can handle per second. The routing algorithm affects the processing time due to its complexity. Each routing algorithm has a time complexity related to the network node number (V) and the connections between these nodes (E), and it is represented as $g(V,E)$. In this paper, we will assume that Dijkstra algorithm is used to find the best route between two hosts. In general, this algorithm has a complexity of $O(V^2)$ and for simplicity, we will consider $g(V,E) = V^2$. The processing time follows the exponential distribution with an average of $\frac{g(V,E)}{P}$. Because the average controller processing rate (μ) is inversely proportional to the processing time, so:

$$\mu = \frac{P}{g(V,E)} \tag{2}$$

3.3 Control Plane Architecture

SDN has three main architectural designs: centralized, decentralized, and hierarchical. Centralized architecture is the initial proposal for the SDN control plane where only one controller is used and stores all the network information. This structure is very simple, but the controller will be a single point of failure when the network devices and hosts are increasing. The decentralized design uses multiple controllers that work together to process the network flows. In this category, we will study two cases:

- Decentralized control plane with a global view: where all the controllers have the same information, and they know everything about the network. This can be useful in reducing the traffic load between controllers but also is resource-consuming (CPU and memory).
- Decentralized control plane with a local view: in this category, each controller has only information about its local network, manages it, and doesn't know about other networks. This design needs more communication between controllers.

A hierarchical structure is a type of decentralized design, but the control plane has two layers, leaf, and root. The leaf controller has a local view and manages its network, whereas the root controller is located in the upper layer, and it manages all the leaf controllers and has a global view of the whole network. This hierarchy allows for both centralized control at the top levels and distributed control at the lower levels, so the combination of centralized and distributed control aspects aligns with the fundamental characteristics of hybrid architectures. In the hierarchical model, legacy distributed control mechanism can coexist with SDN controllers, allowing for a smooth transition and integration process. This integration aspect is also a characteristic trait of hybrid architecture. Moreover, the hierarchy model allows for such adaptability and flexibility by enabling different levels of control based on the specific needs of different network segments. This dynamic allocation of control functionalities is a hallmark of hybrid architectures also and asserts that the hierarchical architecture proposed in our study indeed falls within the category of hybrid SDN control plane architecture.

4. CONTROLLER UTILIZATION EVALUATION

4.1 Centralized Design

When there are (h) hosts in the network, each host can communicate with any other host, so there are $h(h-1)$ flows can pass the network and be processed by the controller. Thus, the average arrival and processing rates for flows received by each controller are illustrated in (3) and (4) respectively:

$$\lambda_{centralized} = \lambda h(h-1) \tag{3}$$

$$\mu_{centralized} = \frac{P}{h^2} \tag{4}$$

4.2 Decentralized Design

In decentralized design, the network consists of (c) controllers and (h) hosts, so each controller manages $\frac{h}{c}$ switches and these controllers can be in one of two modes.

- The first mode is a global view mode, all the controllers have the same information about the whole network, and each controller can process any received flow and send it directly to its destination. So, for every controller, the average arrival and processing rates for flows received by each controller are illustrated in (5) and (6) respectively:

$$\lambda_{D,Global} = \lambda \frac{h(h-1)}{c} \tag{5}$$

$$\mu_{D,Global} = \frac{P}{h^2} \tag{6}$$

and the controller utilization is:

$$U_{D,Global} = \frac{\lambda_{D,Global}}{\mu_{D,Global}} \tag{7}$$

- The second mode is a local view mode. Every controller knows only information about its local network. The flow can be local flow if the source and destination belong to the same local network; otherwise, it's a global flow. When the controller receives a local flow, the flow can be processed by that controller simply, but when it's a global view, i.e., the controller has no idea about the destination node, the flow will be divided into two sub-flows, one is a local for the controller itself, and the other is global and will be passed to the neighbor controller for further processing. The neighboring controller will process that flow in the same way until the flow reaches a controller that knows the destination. In the end, the flow is processed by multiple controllers, and we will have a flow hop (γ) that represents the number of controllers needed to process the flow initiation request over the path from the i^{th} host to the j^{th} host, so we can say that each global flow is divided into (γ) local flows. The total flows in the network where there are (c) controllers and (h) hosts can be divided into $\frac{h^2}{c} - h$ local flows and $h^2 - \frac{h^2}{c}$ global flows. So, the average arrival and processing rates for flows received by each controller are illustrated in (8) and (9) respectively:

$$\lambda_{D,Local} = \frac{\lambda}{c}\left[\left(\frac{h^2}{c} - h\right) + \left(h^2 - \frac{h^2}{c}\right) * \gamma\right] \quad (8)$$

$$\mu_{D,Local} = \frac{P}{(h/c)^2} \quad (9)$$

In (8) we multiply the number of global flows by (γ) because each global flow is processed by multiple controllers located on the flow path from source to destination. So, the controller utilization is:

$$U_{D,Local} = \frac{\lambda_{D,Local}}{c * \mu_{D,Local}} \quad (10)$$

4.3 Hierarchical Design

In this architecture, two layers of controllers are introduced, the leaf and the root controllers. Leaf controllers manage only their local networks, and they are abstracted as logical nodes, while the root controller has a global view of the network and it manages the global flows received from leaf controllers. Here we will distinguish between leaf and root controller in terms of average processing rate. Each leaf controller is managing ($\frac{h}{c}$) hosts, while the root controller manages a network with (c) nodes, so its processing rate is:

$$\mu_{H,Root} = \frac{P}{c^2} \quad (11)$$

$$\mu_{H,Leaf} = \frac{P}{(h/c)^2} \quad (12)$$

Similar to the decentralized local view, and because each leaf controller can process both local and global flows whereas the root controller only processes the global flows received by the leaf controllers, the average arrival rates for flows received by each leaf and root controller are illustrated in (13) and (14) respectively:

$$\lambda_{H,Root} = \lambda * (h^2 - \frac{h^2}{c}) \quad (13)$$

$$\lambda_{H,Leaf} = \frac{\lambda}{c}\left[\left(\frac{h^2}{c} - h\right) + \left(h^2 - \frac{h^2}{c}\right) * \gamma\right] \quad (14)$$

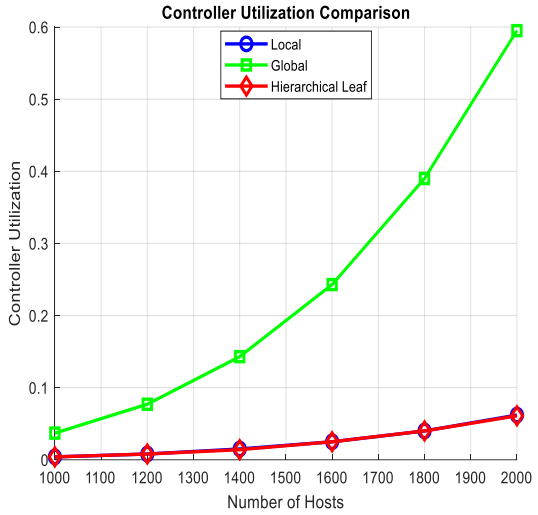
5. RESULTS AND DISCUSSION

In this section, we will have numerical results and will compare the different control plane structures under different conditions. As in (Benson et al., 2010) each host can send (λ=0.001 per second) flow requests to any other host. The experiment was done in a computer with an Intel(R) Core (TM) i5-2450M CPU with 6.00 GB of RAM, and that has a computing power of $P = 2^{30}$ operations per second. The experiment started with the number of hosts h =1000 and increased by 200 until h=2000.

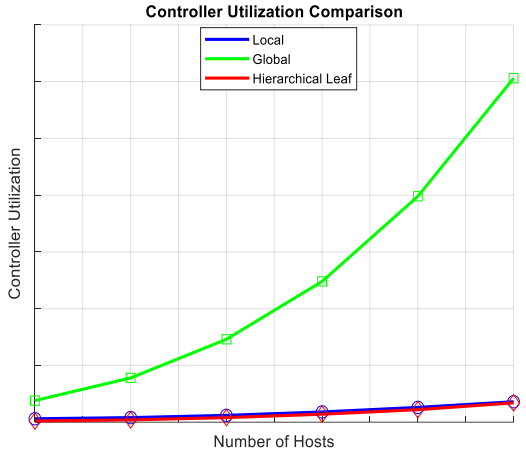
We omit the centralized structure analysis because the controller utilization reaches 93% when the number of hosts is only 1000, and after that, the controller becomes unstable. In decentralized and hierarchal leaf architectures, the controller utilization is directly proportional to the number of hosts but inversely proportional to the number of controllers. The root controller in hierarchal architecture is an exception, as the controller utilization increases when the number of controllers increases, but it is still directly proportional to the number of hosts.

The controller utilization in different architectures is shown in Fig. 2 as we set different controller numbers and different flow hops γ. When c =5 and γ =3, the maximum controller utilization with 2000 hosts is (6%, 5.9%, and 60%) for decentralized local, leaf, and decentralized global architectures, respectively, whereas when c =7 and γ =3, it is (1.6%, 1.6%, and 30%). From Fig. 3, Fig. 4 and Fig. 5, when γ =4 and c =5,7,9, the maximum controller utilization with 2000 hosts in both decentralized local view and hierarchal leaf architectures is (8.1%, 2.2%, and 0.8%), whereas it is (60%, 30%, and 18%) in decentralized global architecture respectively.

We conclude that controller utilization in decentralized local view and hierarchal designs (leaf controllers) are almost the same, and both are better than the global design, which has the worst controller utilization. From Fig. 3 and Fig. 4, we can see that the controller utilization is getting better as the number of controllers is increasing, but for each number of controllers, it is getting worse as γ is increasing due to the increasing number of controllers involved in the flow processing. For example, for decentralized local view architecture, when c =5, the maximum utilization with 2000 hosts is (1.15%, 1.68%, and 2.21%) when γ =2,3,4, respectively. Fig. 5 and Fig. 6 indicate that the root controller has the best utilization, while the decentralized global view design has the worst. In terms of task execution time, the optimal utilization in hierarchical design can directly be translated to efficient task execution time. Within this design, tasks processed demonstrated faster calculation times, reduced I/O delays and minimized transmission times, while in contrast, poor decentralized global design utilization leads to higher workload which directly impacts task execution times and increases I/O delays with slower transmission times. These findings underscore the importance of selecting an appropriate control plane architecture concerning the utilization and task execution efficiency.

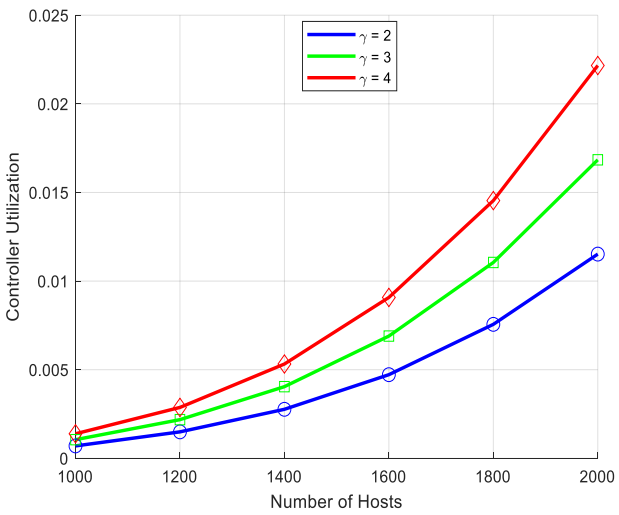


(a). c=5 and γ=3

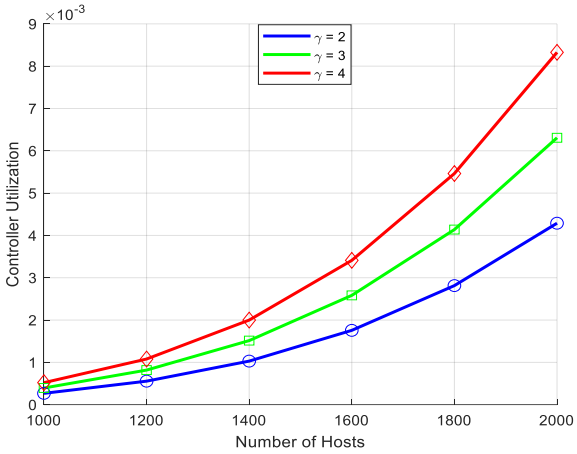


(b). c =7 and γ =3

Fig. 2. Controller utilization in different architectures with different controller and flow hops.

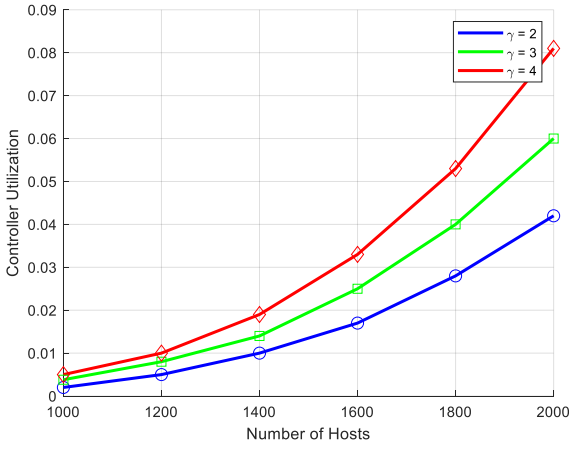


(b) c =7

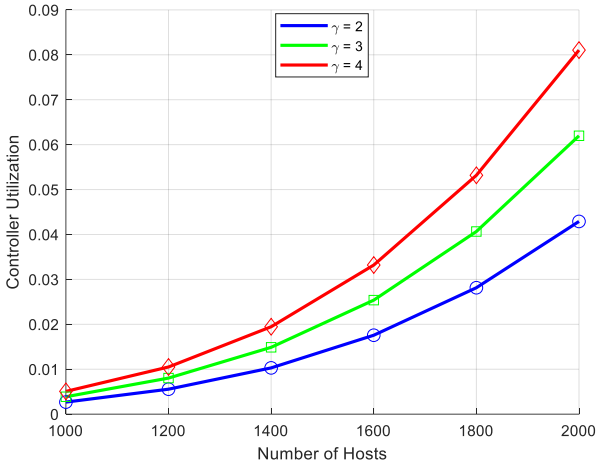


(c) c =9

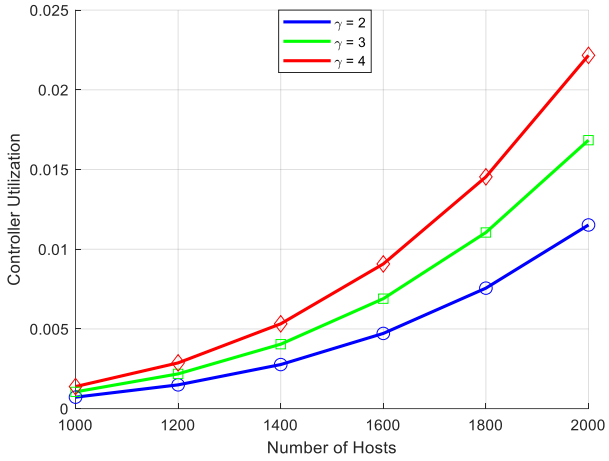
Fig. 3. Decentralized local controller utilization with different controller and flow hops number.



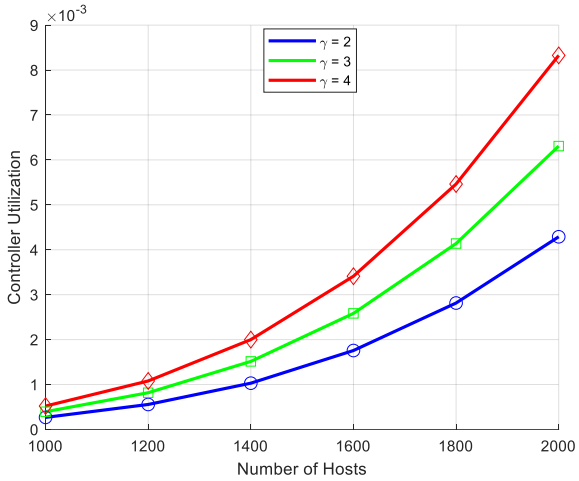
(a) c =5



(a) c =5



(b) c =7



(c) c =9

Fig. 4. Leaf controller utilization with different numbers of controllers and flow hops.

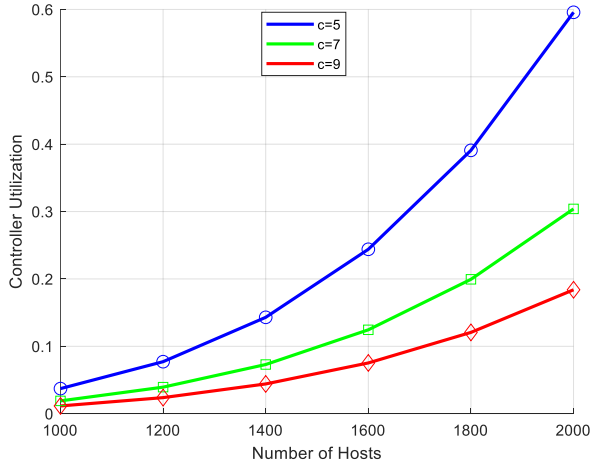


Fig. 5. Global view structure utilization with different numbers of controllers.

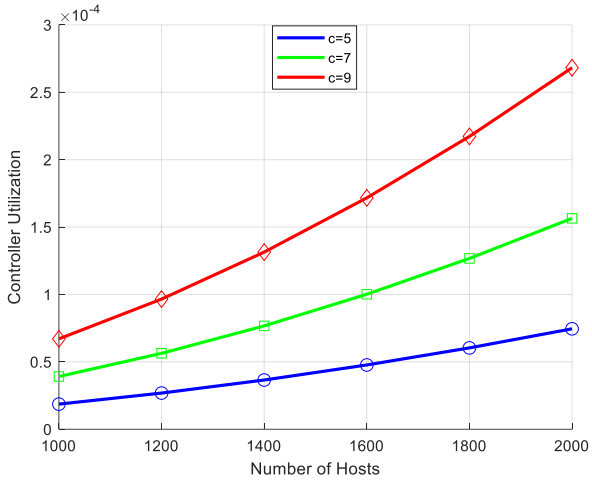


Fig. 6. Root controller utilization with different number of controllers.

6. CONCLUSION AND FUTURE WORKS

In this paper, we study the SDN scalability and focus on control plan utilization in different control plane architectures as a scalability metric. We propose a mathematical model to define controller utilization in decentralized and hierarchal architectures with an increasing number of controllers and hosts. Then, a numerical experiment was done to verify the method. We collect the results when (N=2000), and we conclude that decentralized local and hierarchical leaf controllers have almost the same utilization with (6%, 5.9% when c=5, and 1.6% when c =7) and both outperforming the decentralized global architecture (60% when c =5, and 30% when c =7), while the hierarchical root controller has the best utilization (0.0075%, 0.016%, 0.027% when c =5,7,9 respectively). These findings emphasize the need for careful architectural considerations and have significant implications for network designers and administrators in selecting appropriate SDN architecture. Future works could involve developing load-balancing mechanisms, resource allocation algorithms or traffic engineering technique to improve the scalability of SDN control planes and also, more scalability metrics can be analyzed such as latency and throughput for more understanding of the factors affecting scalability.

REFERENCES

Abdullah, M. Z., Al-Awad, N. A., and Hussein, F. W. (2018). Performance Comparison and Evaluation of Different Software Defined Networks Controllers. *International Journal of Computing and Network Technology*, *6*(2).

Alraawi, A. A. M., and Adam, S. A. N. (n.d.). Performance Evaluation of Controller Based SDN Network Over Non-controller Based Network in Data Center Network. *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 1–4.

Asadollahi, S., Goswami, B., and Sameer, M. (2018). Ryu controller's scalability experiment on software defined networks. *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, 1–5.

Badotra, S., and Panda, S. N. (2020). Experimental comparison and evaluation of various OpenFlow software defined networking controllers. *International Journal of Applied Science and Engineering*, *17*(4), 317–324.

Benson, T., Akella, A., and Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 267–280.

Cabarkapa, D., and Rancic, D. (2022). Software-Defined Networking: The Impact of Scalability on Controller Performance. *2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC)*, 17–21.

Hassas Yeganeh, S., and Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 19–24.

Islam, M. T., Islam, N., and Refat, M. Al. (2020). Node to node performance evaluation through RYU SDN controller. *Wireless Personal Communications*, *112*, 555–570.

Islam, S., Khan, M. A. I., Shorno, S. T., Sarker, S., and Siddik, M. A. (2019). Performance evaluation of SDN controllers in wireless network. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 1–5.

Jogalekar, P., and Woodside, M. (2000). Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, *11*(6), 589–603.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., and Hama, T. (2010). Onix: A distributed control platform for large-scale production networks. *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*.

Lunagariya, D., and Goswami, B. (2021). A comparative performance analysis of stellar sdn controllers using emulators. *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 1–9.

Othman, W. M., Chen, H., Al-Moalmi, A., and Hadi, A. N. (2017). Implementation and performance analysis of SDN firewall on POX controller. *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, 1461–1466.

Ross, S. M. (2014). *Introduction to probability models*. Academic press.

Satre, S. M., Patil, N. S., Khot, S. V, and Saroj, A. A. (2021). Network performance evaluation in software-defined networking. *Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020, Volume 2*, 633–645.

Schmid, S., and Suomela, J. (2013). Exploiting locality in distributed SDN control. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 121–126.

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36–43.

Tootoonchian, A., and Ganjali, Y. (2010). Hyperflow: A distributed control plane for openflow. *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, *3*, 10.5555.

Urrea, C., and Benítez, D. (2021). Software-defined networking solutions, architecture and controllers for the industrial internet of things: A review. *Sensors*, *21*(19), 6585.

Yeganeh, S. H., Tootoonchian, A., and Ganjali, Y. (2013). On scalability of software-defined networking. *IEEE Communications Magazine*, *51*(2), 136–141.

Yurekten, O., and Demirci, M. (2021). SDN-based cyber defense: A survey. *Future Generation Computer Systems*, *115*, 126–149.

Zhou, Q., Zhao, T., Chen, X., Zhong, Y., and Luo, H. (2022). A Fault-Tolerant Transmission Scheme in SDN-Based Industrial IoT (IIoT) over Fiber-Wireless Networks. *Entropy*, *24*(2), 157.