Journal of Computer Science

Original Research Paper

A Scalable Hierarchal Based Addressing Routing Architecture in Software Defined Network (SDN)

Ahmed Gaber Abu Abd-Allah, Ashraf Zaki Ghalwash and Aya Sedky Adly

Department of Computer Science, Helwan University, Egypt

Article history
Received: 12-06-2020
Revised: 21-08-2020
Accepted: 02-09-2020

Corresponding Author:
Ahmed Gaber Abu Abd-Allah
Department of Computer Science, Helwan University, Egypt
Email: ahmedgaber_cs@hotmail.com

Abstract: Software Defined Networking (SDN) designs are initiated with their own apprehensions and challenges that are needed to be addressed by researchers. A Controller, the main and the dominant network resource in SDN, has complex connections that lead to traffic overhead between other devices in the network and also from and to other Controllers. A Controller's capacity of communications obtains will be grown when the network gets bigger. Controller scalability one of the biggest issues of SDN caused by this growth. This paper presents SDN with a new framework architecture based on addressing routing through multiple layers of controllers positioned and encompassed in two levels hierarchal sequent. As a result, an efficient resource allocation and scalability strategy are achieved. The framework is implemented and evaluated. Experimental results show their superiority by reducing the number of messages handled via the Super Intend Controller for each domain and sustained performance for the entire network.

Keywords: Controller Scalability, Hierarchal Framework, Multiple Controllers

Introduction and Motivation

Scalability and Flexibility are common catchy words in the data centers field (Singha *et al*., 2014). Organizations have to seek a cheap way to expand their networks rapidly, whereas monitoring is easy and organizations do not stick to special vendors. Outdated networking has many restrictions that prevent it from meeting the needs of today's users and enterprises due to its limited capabilities. Forming or setting up network strategies and applications needs more skilled people, large budget and time consumption to achieve sustainable network performance. Moreover, in order to achieve scalability in SDN, you will face issues resulting from the centralized controller, which is one of the core ideas of SDN. Many researches try to improve the scalability, especially in the control plane because it is the most complicated part in the network and it is responsible for all communications; and improving scalability comes in particular through reducing the overhead of the centralized controller in various aspects (Karakus and Durresi, 2015; Farhady *et al*., 2015; Tavakoli *et al*., 2009). SDN has been established upon a potential design that includes all the control services in a centralized controller. These services permit a complete wide view of the network, developing control applications a well as applying policies to develop much informal in this setup. However, controllers still can possibly be the tailback in the network process, when the network magnitude raises and more requests and events are sent to the controller, consequently, at some point controller cannot handgrip all the incoming requests (Tavakoli *et al*., 2009). Huge data center means millions of flows and links between network devices and controller, this leads to possible traffic jam and overloaded controller. Benchmarks on installed NOX controllers show that an individual controller can handle at least 30 K new flows setup per second. Decoupling of control plane and data plane, magnitude of requests/events controlled by the controller and switch/controller communication interruption are the reasons leading to the scalability issue (Farhady *et al*., 2015; Tavakoli *et al*., 2009; Rana *et al*., 2019; Metzler and Metzler, 2015; Karakus and Durresi, 2016). Thus, when you think how to reduce the traffic load on the main controller and how to have more than one controller in a hierarchical structure (Oktian *et al*., 2017), where each controller must have its responsibilities, an answer comes to mind. Each controller will be responsible for a sub tree (domain) from the network; meanwhile the network will be divided into certain levels, each one of which has its network addressing. The paper is

© 2020 Ahmed Gaber Abu Abd-Allah, Ashraf Zaki Ghalwash and Aya Sedky Adly. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

organized as follows: Section 2 has the related works done to enhance the controller performance for better scalability. Section 3 explains the details of the proposed framework and how it works and section 4 has the experimental results and evaluations. As well as a proof of concept for the evaluation, comparison and discussion, that shows that the proposed framework outperformer's similar approaches found in the literature. Finally, section 5 summarizes the conclusions and future works.

Related Works

Scalability is one of the dominant SDN challenges (Rana *et al.*, 2019). The objective of scalable Controller is diverse from the network and application's interpretation. The anticipated resolutions to controller scalability problem of an SDN system can be off the record in two comprehensive categories (Metzler and Metzler, 2015). First, switch plane itself is to redefine the controller structure and its hierarchy. Second grouping purposes to adventure some renowned optimization techniques in order to improve the usage of all supportive functions of the resources in SDN, such as data plane resources and management plane resources. To improve the scalability of an SDN network, networking topologies like central controller architecture and scattered networking are famous settings for SDN networks.

As (Karakus and Durresi, 2015) proposed a construction consisted of certain stages from bottom to up, these stages could be increased. The framework proposed in this study consists of two levels: Network Level (bottom level)-Data Plane, consists of independent fields such as ISPs, independent systems which are also SDN domains with their own local controllers; and Broker level (up level)-Control Plane, containing of a super controller substitute like a superintendent for the bottom level controllers. Also an obtainable a categorized SDN architecture and inter-AS QoS based routing methodology, which improved the scalability of the control plane in an SDN network by dropping the number of messages received by controller. A comparison has been placed between the distributed and hierarchal frameworks, ensuring that the hierarchal is more effective than the distributed in different aspects like using the QoS or sharing information between the local controllers. Furthermore, this work showed that the network controller would handgrip less transportations for inter-AS traffic flow in a hierarchic atmosphere compared to non-hierarchic environment since they did not need to keep global network interpretation and coordinate with other situations. This situation reduced the number of messages but the influences increased in the system. Compared to our proposed framework, we have added the factor of the addressing and increasing the responsibility of the local controller to handle more messages. In the proposed framework, the SC will interfere only when we have packets transmission from different domain and different addressing level.

The key factor in the related work was designed to enable network machinists to reproduce resident controllers on call and release the consignment on the upper level, the processes were frequent events in highly replicated local control applications and rare events in a central location. The major issue was that local controllers did not propagate an OpenFlow event unless the root controller had subscribed to that event. Thus, without subscription to all OpenFlow events in all local controllers, we could not guarantee that existing OpenFlow applications work as expected (Hassas Yeganeh and Ganjali, 2012). OpenFlow made the controller manage too many micro-flows, which created extreme load and overhead on the controller and switches. DevoFlow treated with short flows at the OpenFlow switch, while only large flows were directed to the controller to be handled. In addition, this came through two major functions; (a) reducing the need to transfer statistics for small flows and (b) possibly reducing the need to appeal the control plane for maximum flow settings (Curtis *et al.*, 2011).

Proposed Framework

This section explains the details of the proposed framework, its scenarios and how it works.

Framework Architecture

In this section, we are going to propose a framework that depends on dividing the network into certain sub- B tree networks vertically as domains. Each domain has certain number of switches, as shown in Fig. 1. The entire network has its Super Intend Controller that will have the global view of the entire network and this Super Intend Controller is located at level zero (assuming the network from up to down in levels according to the Controller type), we will have two types of controllers, the mentioned above (SC) plus the Domain Controller (DC). In our framework, the levels are uniquely distinguished by network addresses and the Controllers verifies the addresses in case of packets modifying-and this is not in paper scope-(checking where the packets are coming from which level). Next, we are going to demonstrate all possible scenarios of sending and receiving a data packet between the source and destination using the proposed framework and evaluate the interference of all Controllers in the mentioned scenarios.

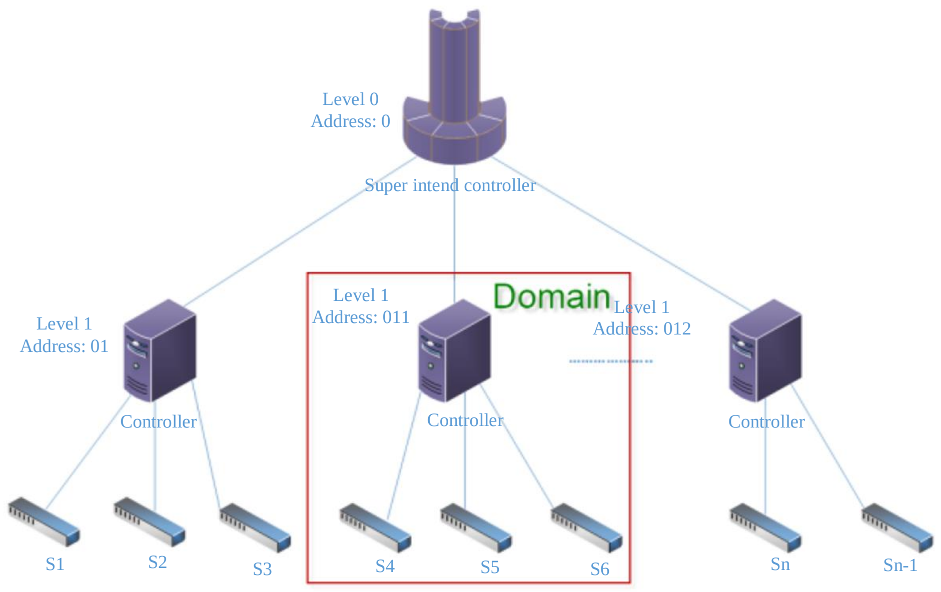


Fig. 1: Basic structure of the proposed framework just shows the types of Controllers and network's levels

Routing Strategies

In this section, the scenarios will be defined and stated below.

Scenario A

As stated in Fig. 2, if we have a source X that wants to send a packet to a destination Y and both are linked to switches under the same domain, this will be the simplest case we could have. The Domain Controller (DC) is aware of all paths inside its cluster or domain, which means that it has a view of all possible paths between the cluster switches, so as stated below there is no way for a connection to be settled between the DC and the SC.

The packet will directly be delivered to the destination upon the flow entries defined by the DC inside the flow table in each switch controlled by the mentioned DC.

Scenario B

When we have a sender X needs to send packet to a receiver Y, as per Fig. 3 (different domain but in the same address level). Moreover, in such case, getting back to the SC still suspended, the only way to figure out the needed and optimal path for the packet transmission in case the neighbor DC is not connected or down for any reasons because if the neighbor DC is available and connected to the requester DC, then there is no need to the SC.

In this example, X that is connected to Switch (S1) wants to send a packet to Y which is connected to Switch (S6), so each DC is aware of its switches paths and has a map view of its domain which is shared with other DC(s) of the same level. In such case the DC must get back to the neighbor DC(s) in its level, sending control packets asking for the optimal path.

In this case, the DC declares a specific switch as border node and this happens through a declaration process of all paths done by each DC (declaration process is not included in this study scope), we demonstrate the transferring of packets through the border nodes in Fig. 5 and 6.

Scenario C

The most complicated case, when X sends to Y and both are in different domain and level (Fig. 4). In this case, the DC from different levels cannot talk to each other's. When the DC cannot offer the needed path, it will get back to the SC, sending a message asking for the optimal path. The SC replying with the right path and the DC(s) will use their border nodes as mention above in scenario B.

As stated in Fig. 5, the red dot line clarifies that the process is hold until getting the right path to Y through sending a message to the SC asking for it, because the SC has the global network map view. Therefore, each connection is a link between source and destination and each of them is in a different domain, then each connection equals two messages sent by the SC.

S3 and S4 act as borders nodes of two different domains with different addresses (Fig. 4 to 6). The DC of each domain distinguises these addresses because it has the local map view of its domain. Therefore, to determine which domain in our proposed framework the packet belongs to, 3-bits will be reserved as IP header the packet is captured at the forwarding device waiting for a confirmation from the DC after the forwarding device checks and finds no match in the flow table.

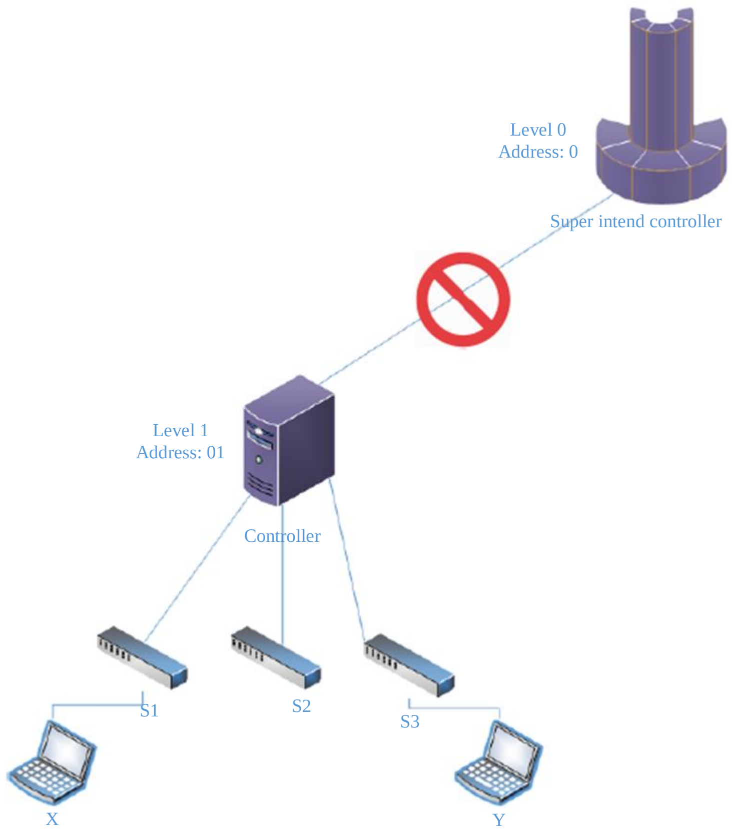


Fig. 2: X sends a packet to Y in the same domain and the communication to the Super Intend Controller is not allowed

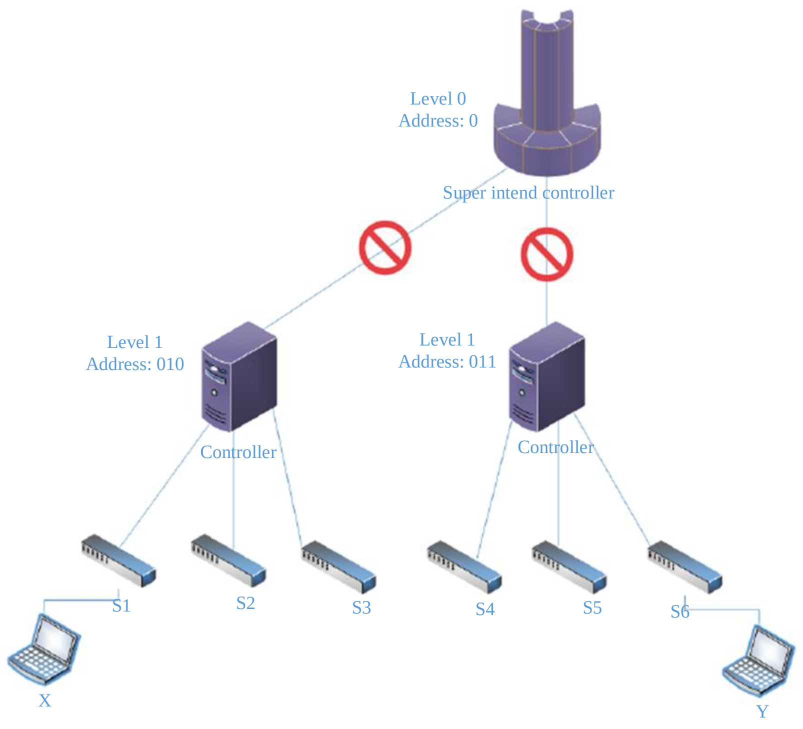


Fig. 3: X sends packet to Y - different domain and both requester and neighbor DCs are not connected

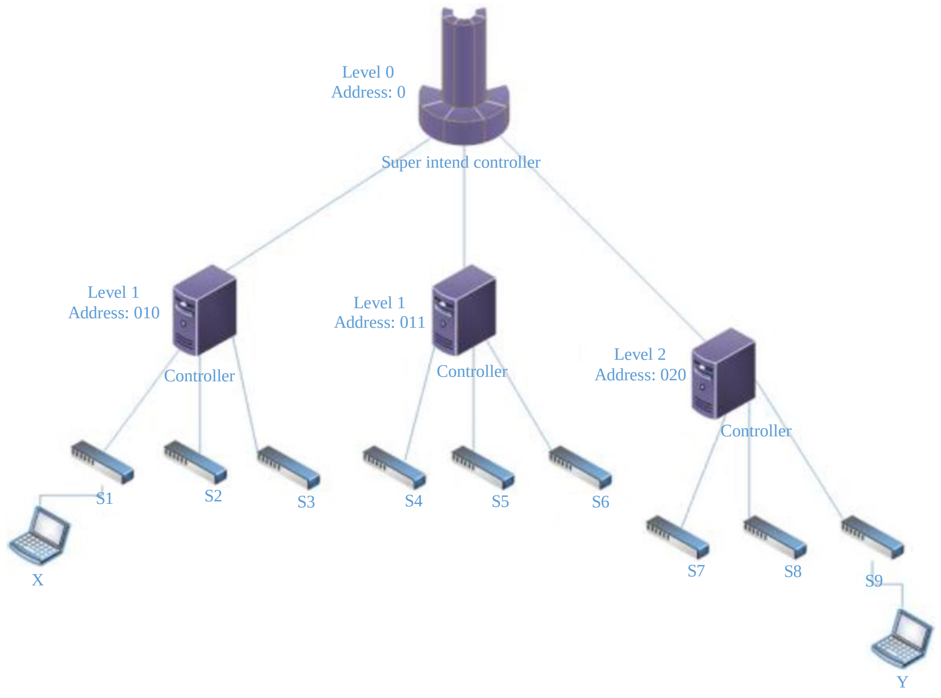


Fig. 4: Scenario C, the multi-level view for the proposed framework

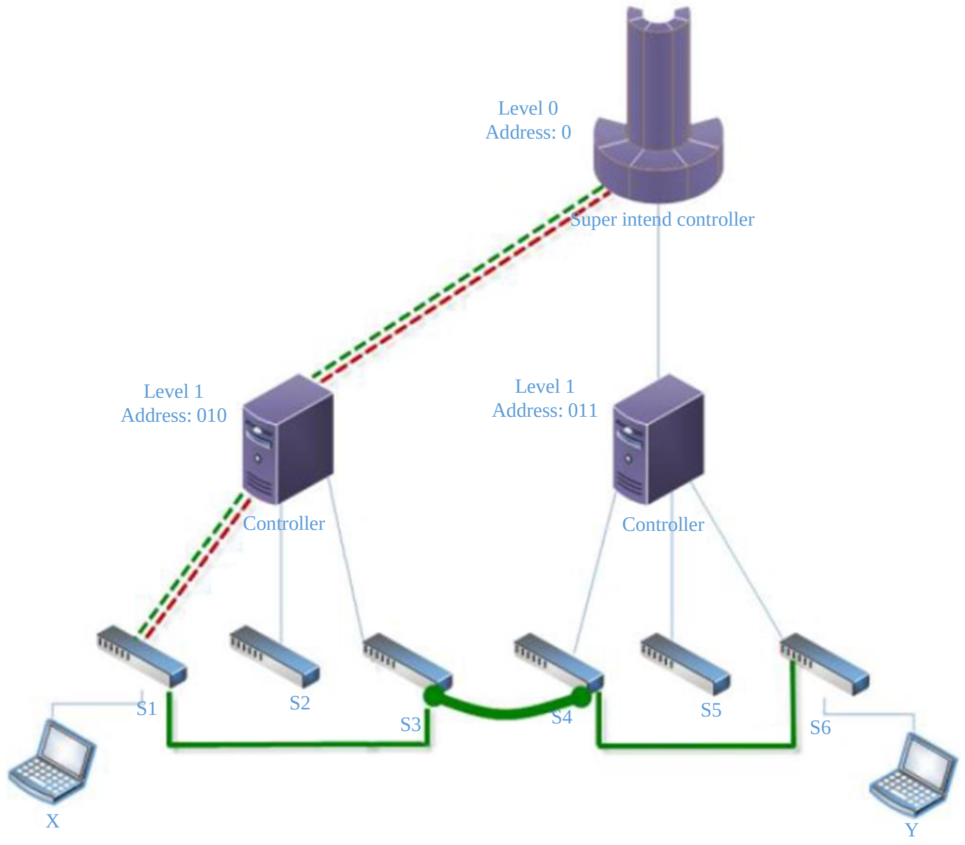


Fig. 5: X sends a packet to Y using S3 and S4 as border nodes, different domains and levels

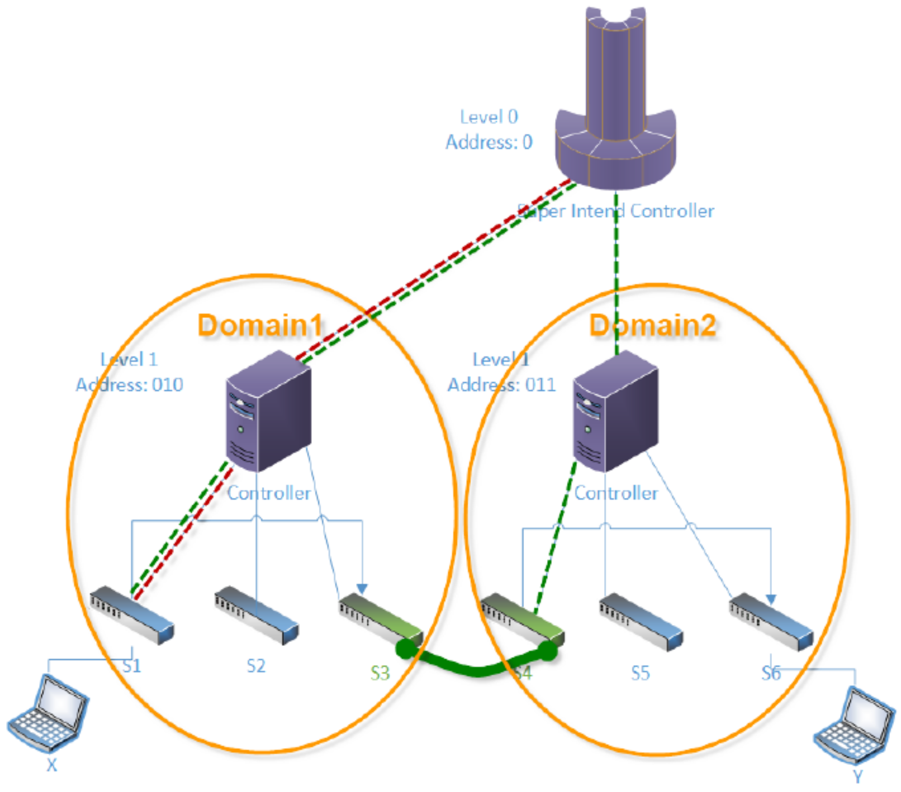


Fig. 6: X sends a packet to Y, different domains but same addressing level using domain border nodes

Then, the DC checks the map view of its domain; if the packet received by the switch belongs to its domain range of addresses, then the DC will send an event to the switch to add that entry in the flow table (Table 1) because it is supposed to be listed (packet held for modification process). Otherwise, the DC will send a message to the SC requesting a path to the destination to reply to the switch with the requested path. As shown below, S3 will forward the packet received from S1 to S4 and then S4 forwards the packet to S6 then to the destination Y.

The DC must update S1 by the new destination path and refresh its flow table by adding the new entry as shown in Table 1 (Updating the header fields, actions and statistics attributes) (Open Networking Foundation, 2015).

Figure 6 clarifies the dual connection happened between the SC, DC1 and DC2 by sending messages (Fig. 6), the first message from DC1 is requesting the interference of SC through replying to DC1 by the requested path. The second message has been sent to DC2 requesting to run the declaration process and get the border node ready. In addition, DC2 refreshes the flow table of its switches by the new destination.

In such case, each DC will declare the border node that will be responsible for receiving and forwarding from and to different domains and also share its network view to the SC, so the SC could synchronize the network map view to each DC in the entire network. According to the decelerated borders of each domain, we have in that case two different domains (and different address level) with two borders that will be responsible for delivering the packet, according to the path obtained from the SC. The SC will be invoked; it will be much less headache over the SC because that path will be recorded for repeated cases. In other words, the SC will be invoked in different domains and different levels with declaring different border nodes.

After detecting that the destination is not in the same domain, the DC will send message to the SC that will have the capability of detecting the address of the destination attached to the packet bit header (3-bits). The SC to all DC already synchronizes this information and it will be used to determine the maximum number of switches the domain should obtain (not in paper scope).

Since the domain of the destination is different, the DC must forward a request to the SC to interfere and then the SC will reply with the needed and optimal path to the destination. In Fig. 6, switches (S3 and S4) represent the domains border nodes. S3 is the domain border node of domain 1 and S4 is the border node of domain 2. As highlighted in Fig. 6 the path will be S1 $\rightarrow$ S3 $\rightarrow$ S4 $\rightarrow$ S6.

Table 1: Insertion of flow entry in the flow table in a switch

Flow entry 1		Flow entry 2		Flow entry N	
Header fields	PortIn&Out VLAN ID IP Port#..	Header fields	PortIn&Out VLAN ID IP Port#..	Header fields	PortIn&Out VLAN ID IP Port#..
Actions	Val	Actions	Val	Actions	Val
Statistics	Val	Statistics	Val	Statistics	Val

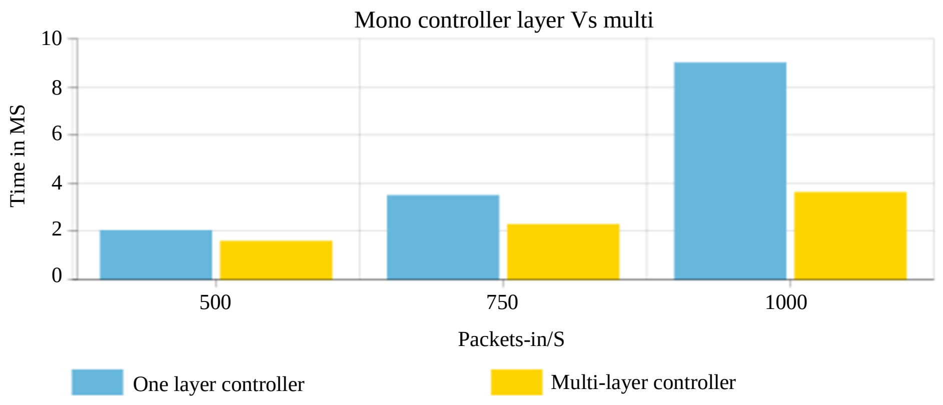


Fig. 7: Latency for packet processing in the proposed model using ONOS from one host to another

Experimental Results and Evaluation

In this section, we will show the obtained latency time and the pinging rate for the proposed model. We built our results through creating our network topology by using Mininet (Mininet, 2018), forwarding devices like OpenvSwitch, which is multilayer virtual switch licensed under the open source Apache 2.0 and Open Networking Operating System (ONOS) version 2.0 (ONF, 2020), which had been used as a controller in the topology with multi controller feature to represent the proposed topology on it. The test packets were captured many times by using capturing applications installed on the ONOS (northbound area), like packet generator, to ensure our results. As shown in Fig. 7, the latency was much better in multi-layer controllers compared to normal state of any mono controller SDN topology. In our topology, we set the ONOS as the SC and the other DCs were created within the Mininet.

The test topology consisted of two levels of controllers with four controllers and three openvSwitches connected to each DC and all controllers are connected to the SC. Finally, the sender and receiver were presented in two Virtual PC Simulator (VPCS), which allowed you to simulate a frivolous PC supporting DHCP and ping features. Furthermore, we did that latency test from specific host to another and vice versa (Fig. 8 to 10) in different domain (Scenario B) in two cases: Case 1; activating the SC, which came out with better results than case 2; deactivating it (normal SDN status).

We evaluated the number of network actions handled by the SC (Control Plane) of the scenario B in our proposed hierarchic architecture. Meanwhile, the aim was to decrease the number of messages and actions the controller's exchanges. We used generated packet of 64 bytes and sent it twice through two hosts connected to different domains; firstly, without using the SC (Fig. 11) and secondly, with activating the proposed topology (Fig. 12).

The connections and messages are two important factors. The number of connections means each link from forwarding device to another and the number of messages is the number of messages sent from and to the SC. When the source and destination are located in different domains and different levels, a message must be sent from the sender DC to the SC and then the SC must send messages replying back to the sender and destination DCs.

Consequently, the interference of the SC increases and, thus, the number of messages increases; this relation is a directly proportional relationship. Therefore, when we need to eliminate that interference, we must decrease the number of messages. So one of our aims is represented in increasing the number of nodes in each domain or cluster. In our framework, we transformed the cluster into two level hierarchal form. The probability of sending and receiving without SC (between the domains in the same level) will be increased. Then the relationship between increasing scalability and interference of the SC is inversely proportional.

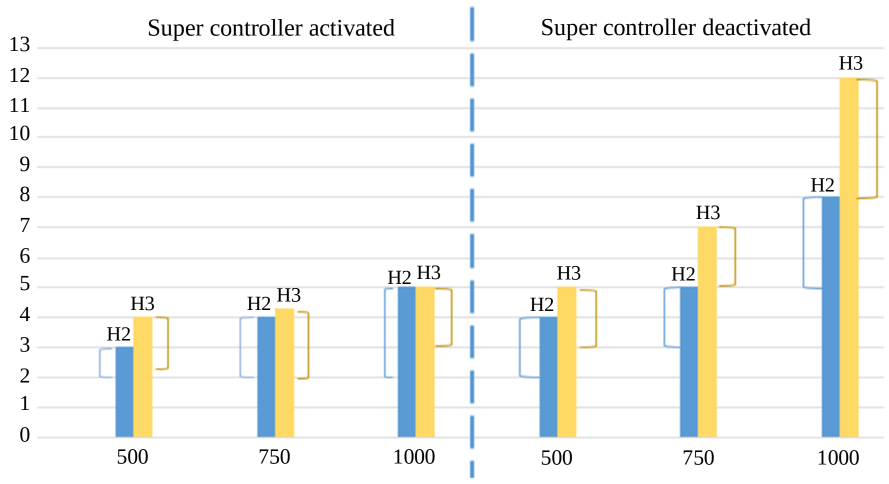


Fig. 8: Host1 sends to Host2 and Host3 in the two cases of activating and deactivating the SC with the same packets number

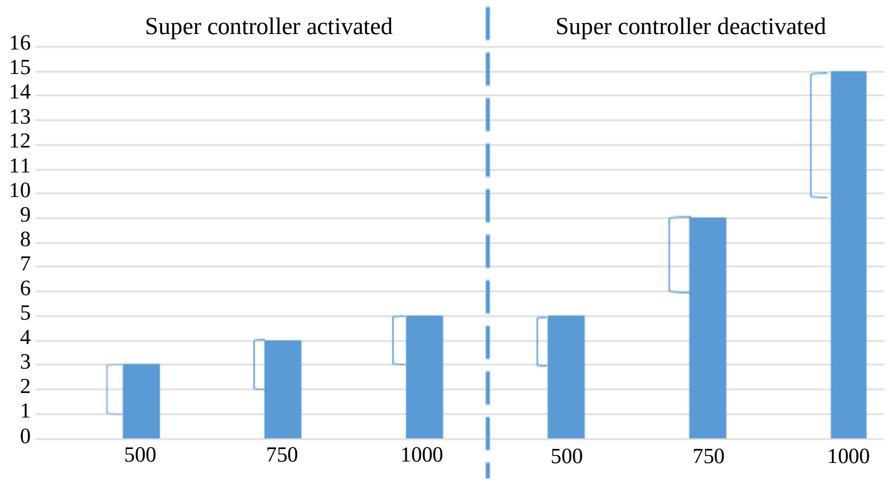


Fig. 9: Host2 sends to Host1 in two cases of activating and deactivating the SC with the same packets number

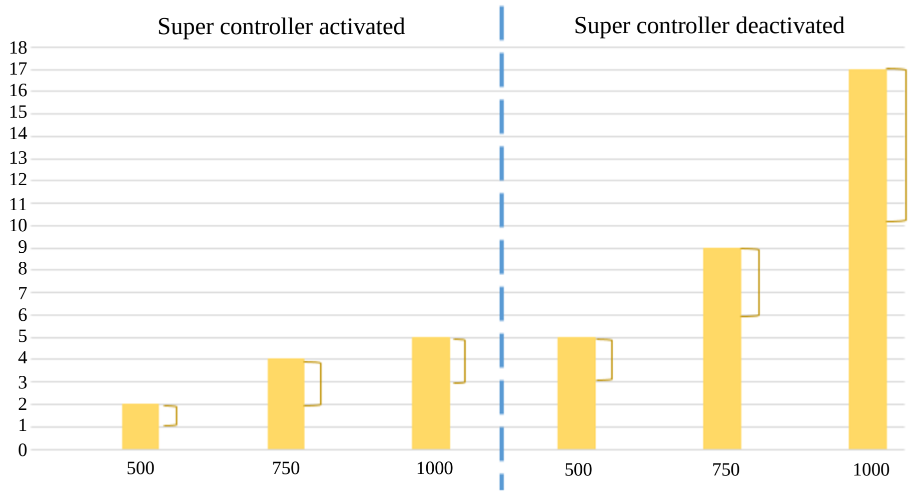
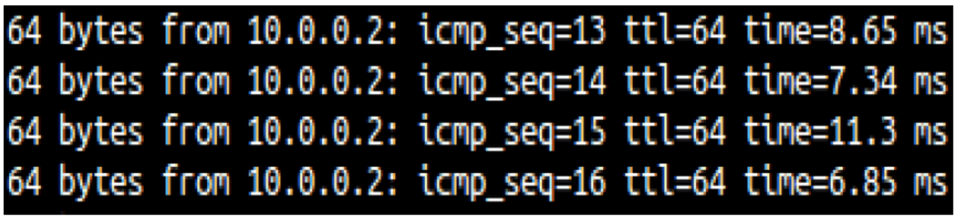
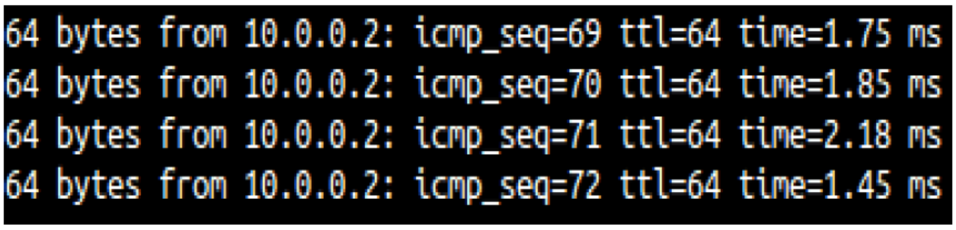


Fig. 10: Host3 sends to Host1 in two cases of activating and deactivating the SC with the same packets number



```
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=8.65 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=7.34 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=11.3 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=6.85 ms
```

Fig. 11: Pinging results after disconnecting the SC



```
64 bytes from 10.0.0.2: icmp_seq=69 ttl=64 time=1.75 ms
64 bytes from 10.0.0.2: icmp_seq=70 ttl=64 time=1.85 ms
64 bytes from 10.0.0.2: icmp_seq=71 ttl=64 time=2.18 ms
64 bytes from 10.0.0.2: icmp_seq=72 ttl=64 time=1.45 ms
```

Fig. 12: Pinging results after activating the SC

Conclusion and Future Works

This paper offered a hierarchical SDN structural design and addressed an established transmitting methodology. Enhancement of scalability was the aim in the offered architecture through fine-tuning the control plane (reallocation and expanding of controllers) in an SDN network by decreasing the number of messages that a network main controller can handle. The only case could the DC can communicate with the SC when we have sender and receiver from different levels. The experimental results showed that the proposed network with super controller handled less messages than those discussed in the literature with very good latency time and sustained performance. Moving forward to the deep north bound area in SDN; we are encompassing our proposed framework to deliver new classes of control applications. Such applications can operate by having access to the events generated by the DC and for advanced level by the switch too.

Acknowledgement

Authors would like to thank editors and all reviewers for appreciated comments.

Author's Contributions

Ahmed G. AbuAbdallah: Conceptualization, methodology, software, visualization and virtualization, data creation and writing-original draft preparation.

Atef Z. Ghalwash: Conceptualization, Supervision and Investigation.

Aya S. Adly: Writing-reviewing and editing.

The final manuscript has been approved by all authors. All authors have read and approved the final manuscript.

Ethics

This paper is original and innovative and it contains unpublished material. There are no ethical issues involved and all authors have no conflicts of interest to release.

References

Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., & Banerjee, S. (2011, August). DevoFlow: Scaling flow management for high-performance networks. In Proceedings of the ACM SIGCOMM 2011 conference (pp. 254-265).

Farhady, H., Lee, H., & Nakao, A. (2015). Software-defined networking: A survey. Computer Networks, 81, 79-95.

Hassas Yeganeh, S., & Ganjali, Y. (2012, August). Kandoo: A framework for efficient and scalable offloading of control applications. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 19-24).

Karakus, M., & Durresi, A. (2015, March). A scalable inter-as qos routing architecture in software defined network (sdn). In 2015 IEEE 29th International Conference on Advanced Information Networking and Applications (pp. 148-154). IEEE.

Karakus, M., & Durresi, A. (2016). A survey: Control plane scalability issues and approaches in software-defined networking (SDN). Computer Networks, 112, 279-293.

Metzler, A., & Metzler, A. (2015). Ten Things to Look for in an SDN Controller (p. 11). Technical Report, May.

Mininet, 2018. http://mininet.org/

Oktian, Y. E., Lee, S., Lee, H., & Lam, J. (2017). Distributed SDN controller system: A survey on design choice. computer networks, 121, 100-111.

ONF, 2020. Open Networking Foundation. https://www.opennetworking.org/onos/

Open Networking Foundation. (2015) Version 1.5.1. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

Rana, D. S., Dhondiyal, S. A., & Chamoli, S. K. (2019). Software defined networking (SDN) challenges, issues and solution. Int. J. Comput. Sci. Eng, 7, 1-7.

Singha, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking: State of the art and research challenges. Computer Networks, 72, 74-98.

Tavakoli, A., Casado, M., Koponen, T., & Shenker, S. (2009, October). Applying NOX to the Datacenter. In HotNets.