Load Balancing in Software-Defined Networking using Controller Placement

Hamid Nejadnik
IAUDA
Rasool Sadeghi (r.sadeghi.2005@gmail.com)
IAUDA
Sayed Mahdi Faghih Imani
IAUDA

Research

Keywords: SDN, Load balancing, Controller Placement, OFSwitch13

Posted Date: August 10th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-53407/v1

Load Balancing in Software-Defined Networking using Controller Placement

Hamid Nejadnik[1], Rasool Sadeghi[2*], Sayed Mahdi Faghih Imani[1]
[1]Department of Computer Engineering, Dolatabad Branch, Islamic Azad University, Isfahan, Iran
[2]Department of Electrical Engineering, Dolatabad Branch, Islamic Azad University, Isfahan, Iran

r.sadeghi.2005@gmail.com(r.sadeghi@iauda.ac.ir), smahdi.imani@googlemail.com, Hamidnejadnick@gmail.com

[*]Corresponding Author

Abstract- Software Defined Networking (SDN) is a novel architecture that separates the data plane from the control plane using an external controller. Similar to traditional networks, load balancing has a great impact on the performance and availability of SDN. Therefore, the Controller Placement Problem (CPP) in SDN influences on the load balancing solutions. In this paper, various topologies of CPP including different load balancer controllers are simulated and evaluated in the SDN using the OFSwitch13 module of ns-3 network simulator. The results provide a solid comparison of the proposed topologies in different network situations.

Keyword- *SDN, Load balancing, Controller Placement, OFSwitch13*

1. Introduction

The rapid evolution of communication technologies and the current demand of the market have posed the importance of more agility and flexibility in network architecture [1]. Software-Defined Networking (SDN) has been emerged as a solution to overcome the limitations. This solution provides the separation of the data plane from the control plane [2]. In SDN, the data plane is assigned to simple devices that perform receiving and sending data packets while the management and routing of packets are transferred to a central point called controller [3-4]. In this architecture with the prominent role of the controller, the issue of load balancing cannot be implemented using the distributed approach of traditional networks. Therefore, the load balancing of SDN can be achieved through the issues related to controllers such as their capabilities, topologies and Placement algorithms [5-6-7]

The load balancing in SDN is obtained through architectural and centralized architecture [6-8-9-10]. In a centralized architecture, a central controller communicates to the switches to collect statistical information to have a general overview of the entire network. Then, the controller sends the associated rules to prevent congestion and to perform the load balancing. [9]. This architecture is not efficient when the network dimensions grow and the number of nodes increases. Therefore, the issue of distributed architecture can address the scalability issue and the need for more controllers [6]. In a distributed architecture, interfaces permit the controllers to share the information needed to manage the entire network. The purpose of this architecture is to eliminate the problem of a single of failure and to improve the network scalability using multiple controllers [10]

Without any load balancing mechanism between servers, the traffic load on a server or a communication link increases significantly leading to network congestion, more delay network, and efficiency degradation [11]. Using load balancing mechanisms, the traffic load is distributed between the servers providing better network performance without congestion on a server [12]. In the following, we review some contributions of the load balancing mechanism in SDN.

In [13], Handigol et al. proposed a load balancing mechanism that minimizes the response time of client requests to the server by reducing the response time of the controller. In another mechanism proposed in [14], the load balancing is obtained through a controller called “plug-n-server”. This controller performs the load balancing between multiple servers using

task dedication and reduction of response time to the switch. Although this mechanism provides high flexibility, it cannot be efficient in scalability and high traffic load. By applying some modification on the "plug-n-server" controller, a new algorithm called "Aster*X" was proposed in [15] which is more scalable while it supports distributed servers. Moreover, this algorithm can be employed in network infrastructure. However, the problem of performance degradation in high traffic load has not been resolved. To solve this problem, the authors of [16] propose an algorithm which reduces the number of switch requests to the controller after efficient rules are installed on the switches. This technique leads to send large groups of clients directly to the server without contacting the controller.

In some load balancing solutions, the issue of load balancing is associated with the controller placement problem (CPP) [7]. Jean et al. [17] proposed different scenarios for controller placement and load balancing in SDN using four parameters: latency, reliability, cost, and multiple objective optimizations. Besides, the controller placement in Open Science, Scholarship and Services Exchange (OS3E) and Zoo networks has been investigated by Heller et al [18]. They investigate transmission delay between controller and switch for different numbers and controllers' locations in various topologies. The performance of different controller's placement was evaluated by measuring the minimum and maximum propagation delays between controllers and their averages value. According to the results for different topologies, adding a controller reduces the delay, and using k controller reduces the delay by 1/k. In dynamic traffic conditions, the lack of proper controller placement may increase the delay in the network. The solution has two folds: optimizing the controller placement and optimizing the controller's dedication time to switches in different traffic conditions [9].

In another research carried out by Chavez et al. [21], a new topology was proposed which exploits a proprietary balancing controller [19]. This controller employs the Round-Robin algorithm [20] and new feature of measurement tables which is supported in OpenFlow 1.3. These capabilities provide load balancing between servers. Figure 1 shows the proposed topology in [21] where two servers and four clients communicate through a client switch, an aggregation switch, and a Boarder switch, and these switches are managed by two controllers: Quality of Service (QoS) and learning. To manage flows and load balance, the border switch sends a request of route selection to the QoS controller. This controller selects one of the servers as the destination and sets the corresponding action in the switch flow tables. Accordingly, the switch forwards the flow to the correct server. This topology is characterized by its ability to be implemented in real network infrastructure and to install efficient rules on switches. However, this topology suffers from high propagation delay and high response time of the controller in high traffic load leading to efficiency degradation of the controller.

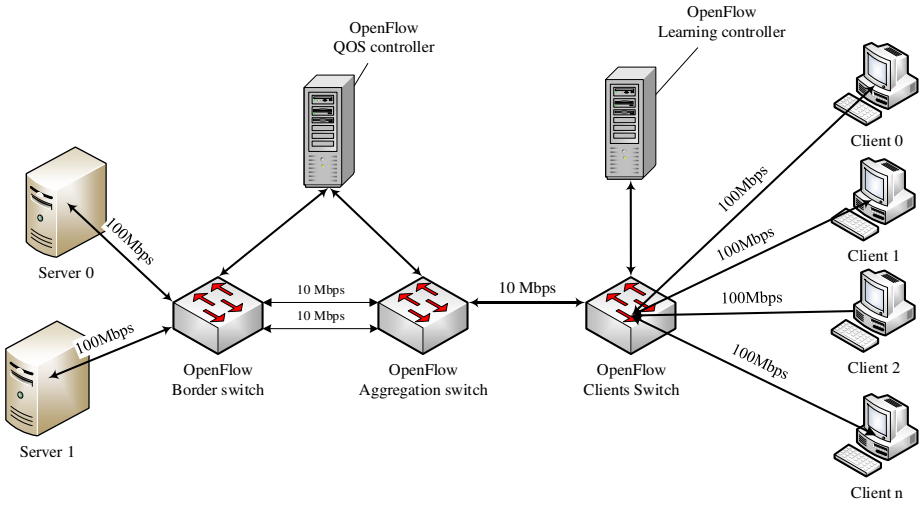


Figure 1: Chavez topology [19]

In this paper, several topologies are proposed according to the capabilities of OFSwitch13 which provide load balancing in SDN by placing different controllers. Moreover, the proposed topologies are evaluated in different scenarios using the ns-3 network simulator. The simulation results provide a comprehensive comparison of the proposed topologies in achieving SDN load balancing.

In the remaining of this paper, the proposed topologies for SDN load balancing with different locations of the controllers are presented in the next section. In Section 3, the simulation scenarios and their results are presented. Finally, Section 4 concludes the paper with some future research lines.

2. Proposed topologies for SDN load balancing

In this section, we modify the topology proposed by Chavez et al. [19], and four new topologies with different controller placement are presented with the aim of load balancing in the SDN. In Chavez topology [19], simultaneous management of border switches and aggregation switches in high traffic load leads to long queues and high delay in the processor of the controller leading to different traffic load on the servers. In the following of this section, the load balancing between the servers is investigated in four scenarios with different controller placements.

2.1 2Q1L topology

The first proposed topology in this section is 2Q1L including two QoS controllers and one learning controller as shown in Figure 2. The main objective of this topology is to solve the problem of sending a large number of requests and a high delay in the controller's response by separating the management of the border switch from the aggregation switch by the controller. It is expected that this separation of switches ' management improves the issue of the load balancing between servers.

2.2 Multiple Controller Topology

One of the new features in Openflow 1.3 is the multiple controller capability which allows managing a switch simultaneously. The main controller is defined as the *Master* and the other controllers would be as the *Slave*. In the case of a Master controller failure, one of the Slave controllers is replaced and the switch is managed. The most important feature of multiple controller topology is its high reliability which the malfunction of one of the controllers does not affect the operation of the entire network or switch.

As shown in Figure 3, two QoS controllers simultaneously manage both the aggregation and border switches. Simultaneous managing of the multiple controllers is used when a large volume of traffic is sent to the switch or one of the controllers is disrupted. In this situation, both switches are managed by the other controller and it prevents the whole network from malfunctioning.

2.3 1Q2L topology

To increase the quality of service to more clients, as well as to increase the reliability of the entire network, 2Q1L topology is proposed including QoS controllers and two learning

controllers (Figure 4). This topology employs two separate learning controllers to control and manage the client switch. If the number of clients increases and new requests arrive, the second controller will be active to manage the flows and help the first controller. In addition, in the case of a malfunction in one of the learning controllers, the performance of the entire topology is not impaired and the reliability will improve.

2.4 2Q2L topology

According to the features proposed in 2Q1L and 1Q2L topologies, a new combination topology called 2Q2L is proposed which includes two QoS controllers and two learning controllers (Figure 5). It is expected that separate management of border and aggregation switches; and management of client switch by two learning controllers provides better performance in comparison to every topology of 2Q1L and 1Q2L.

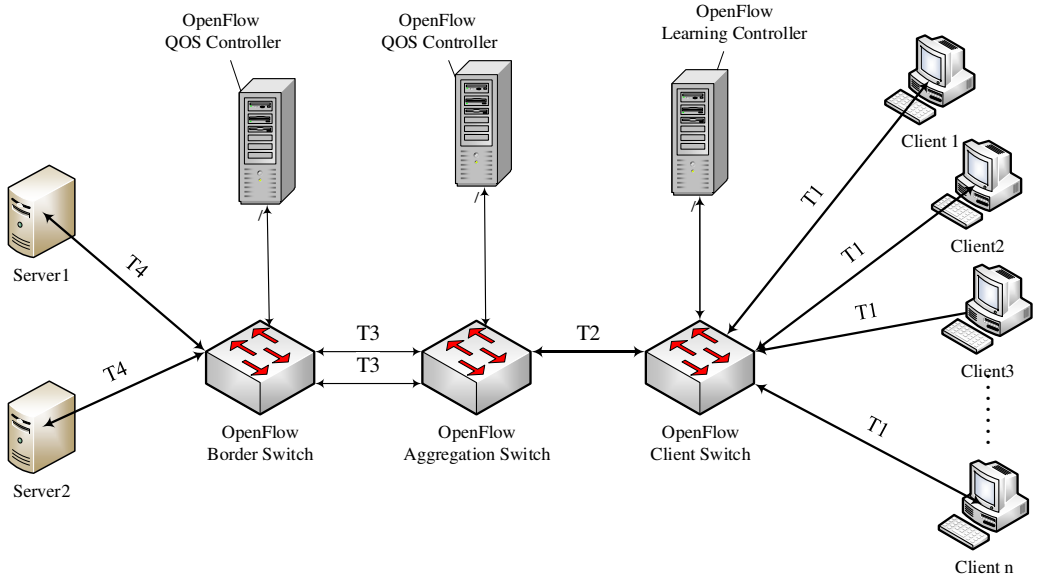


Figure 2: 2Q1L topology

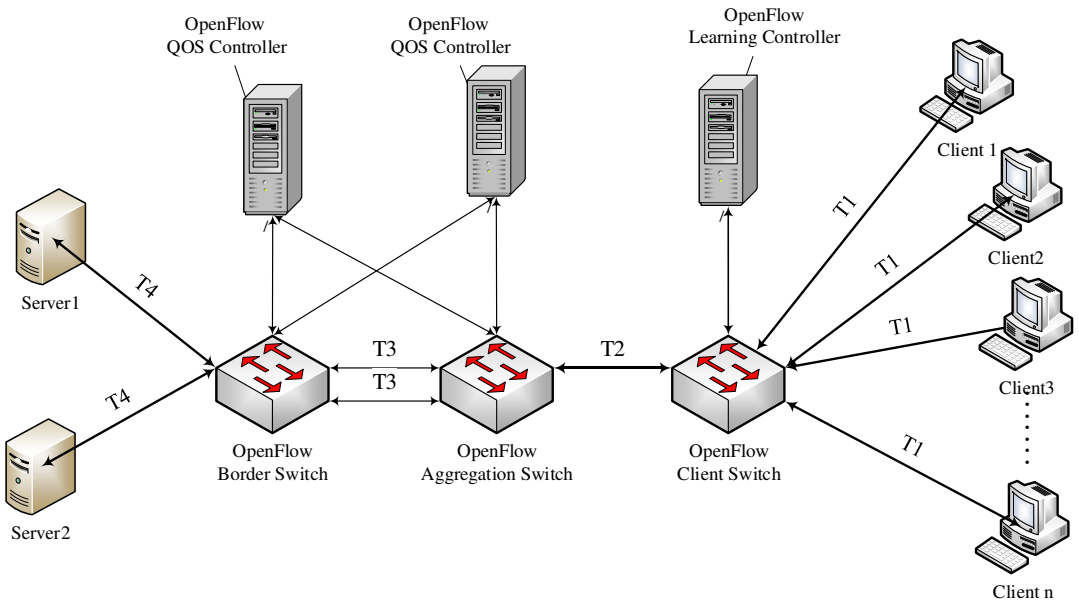


Figure 3: Multiple controller topology

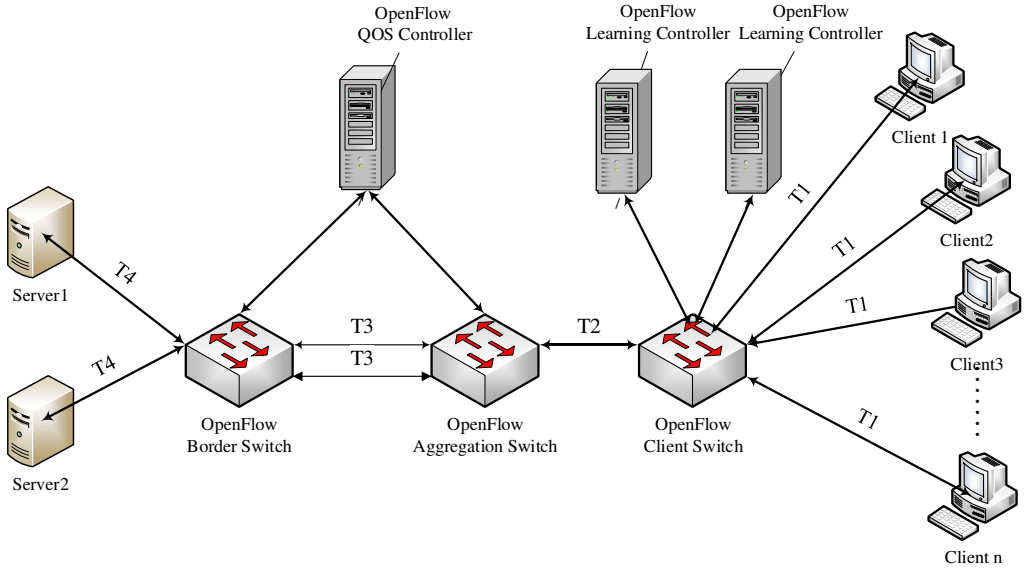


Figure 4: 1Q2L topology

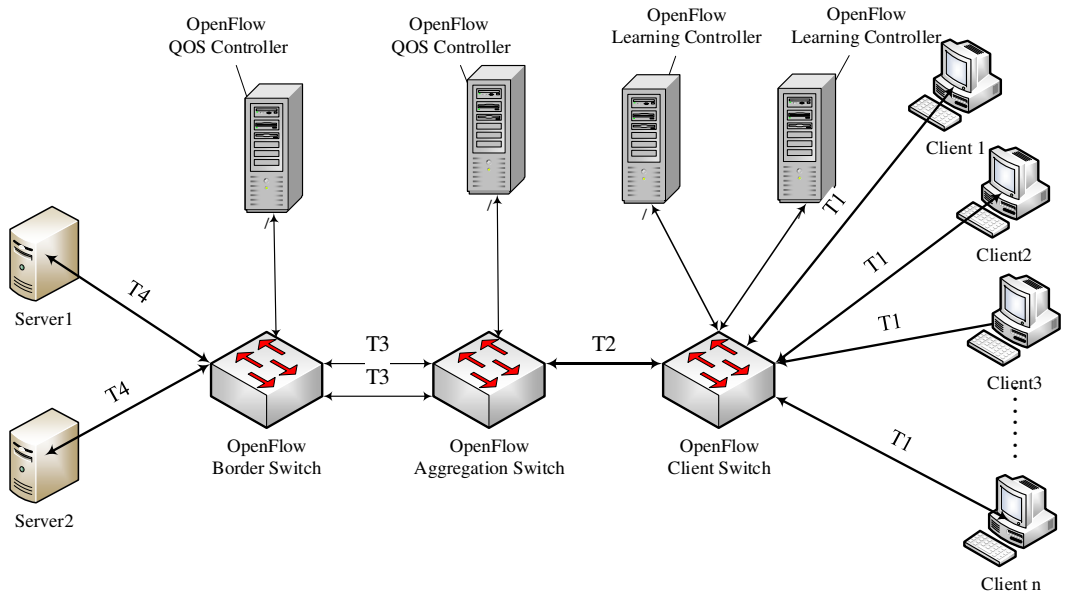


Figure 5: 2Q2L topology

3. Simulation and Evaluation

In this section, we propose two metrics for load balancing evaluation of topologies in Section 2. Then, we will perform simulation in several scenarios and evaluate the topologies based on the proposed metrics. Suppose S_1 and S_2 are the traffic load sending from the border switch to *Server 1* and *Server 2*, respectively, the load difference between two servers can be defined as the following:

$$\Delta S_{12} = |S_1 - S_2| \quad (1)$$

The smaller ΔS_{12}, the better the load balancing is achieved, i.e. $\Delta S_{12} = 0$ indicates full load balancing. However, ΔS_{12} does not indicate accurately the whole incoming traffic to the border switch. For this purpose, the β_{12} metric is defined which is obtained by dividing the traffic load difference between the two servers by throughput of the border switch (S_1+S_2).

$$\beta_{12} = \frac{|S_1 - S_2|}{S_1 + S_2} = \frac{\Delta S_{12}}{S_1 + S_2} \quad (2)$$

In this paper, we use the ns-3 network simulator as a tool for simulation of the proposed topologies discussed in Section 2. The ns-3 is a discrete, free and open-source event-based simulator which has a module called OFSwitch13 for supporting OpenFlow 1.3.

To evaluate the proposed topologies of section 2, we set up several scenarios by changing parameters such as the numbers of flow tables, switch CPU capacity, and the number of clients. Table I indicates the setting parameters. The Per-flow meters in OpenFlow 1.3 provides rate limiting which leads to packet classification, QoS policing, and QoS operation. The aggregation link provides the combination of multiple network connections using the group table to increase the overall throughput. The results are an average of 10 runs in 100 Seconds of simulation time. The packet size is 1400 bytes and the link bandwidth is 100 Mbps for all links except two links between aggregation and border switches which is 10 Mbps.

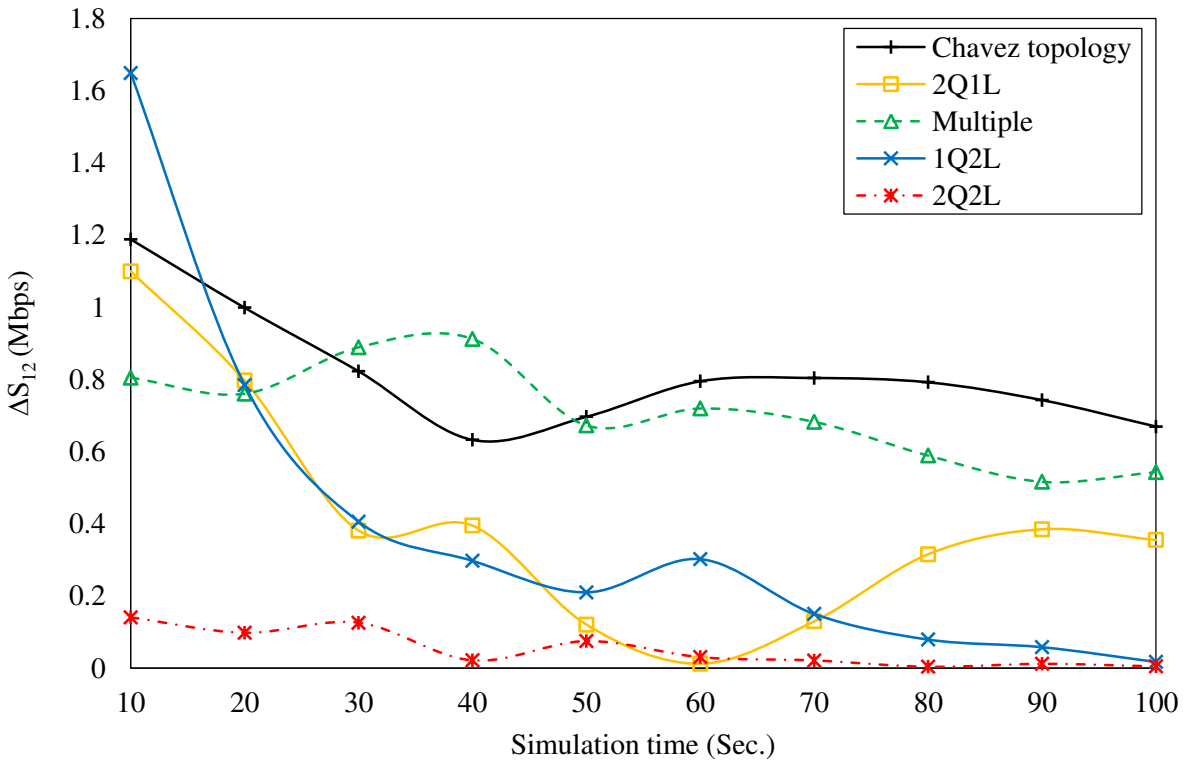
To create a uniform distribution and to prevent simultaneous traffic and high packet loss, a random time between 0 and 1 is assigned to each client to start sending data to the network and it is continued until the end of simulation time. Moreover, to create a saturation scenario, the number of packets sent by clients is unlimited and their packet rate is 512 Pkt/s.

Table I. Simulation parameters

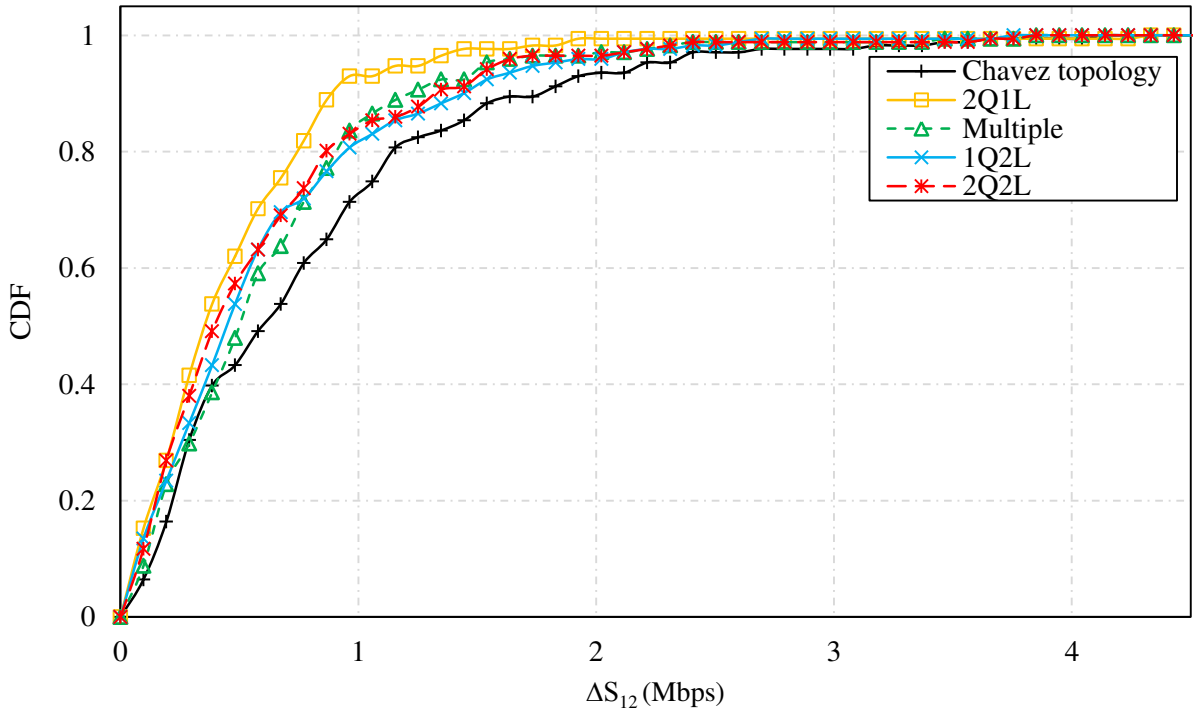
Parameters	Value
PerFlow Meters	True and False
Link aggregation	True
Number of servers	2
Time simulation	100 Seconds
Maximum Transmission Unit	1400 Bytes
Client Packet rate	512 Pkt/s
T1=T2=T4	100 Mbps
T3	10 Mbps

A. ΔS_{12} Measurement and CDF

In the first simulation scenario, all topologies are simulated for 20 clients and other variables are as the default of OFSwitch13 (CPU capacity = 100 Gbps and the number of flow tables=216). Figure 6.a indicates ΔS_{12} during the simulation time. The simulation results demonstrate that better performance in terms of load balancing when compared to Chaves topology. However, the variation of ΔS_{12} value during the simulation time does not permit to have a proper comparison. Therefore, a Cumulative Distribution Function (CDF) of ΔS_{12} value for all topologies is computed to compare all topologies (Figure 6.b). The simulation results in Figure 6.b indicate that the 2Q1L topology provides the best load balancing. As expected, management separation of the border switch from the aggregation switch in the 2Q1L topology leads to the best performance of the load balancing.



(a)



(b)

Figure 6: (a). The ΔS_{12} value for Number of Clients=20, CPU capacity = 100 Gbps and the number of flow tables=216 , (b).Cumulative Distribution Function (CDF) of ΔS_{12}

B. Flow tables variation

In SDN, routing is performed using the flow tables in the switch and matching operations. When a flow arrives at the OpenFlow switch, the header of the packets is first compared to flow table entry. If it is matched, the flow is routed by the switch and the flow is forwarded to the destination. Otherwise, a mismatch occurs and the flow is forwarded to the controller for routing. The problem arises when the number of flow tables is not enough and the switch is constantly forced to send a request to the controller for routing leading to overload on the controller.

The OpenFlow switch tables are usually in the category of expensive ternary content-addressable memory (TCAM), which achieves a single-clock-cycle search time. The capacity of the flow table is limited due to power, cost, and chip size constraints. Therefore, maintaining the flow table size at a reasonable level is especially important to prevent table overflow and maintain switch performance. To evaluate the impact of the flow table's variation on the network performance, different topologies are simulated with a different number of flow tables. The number of flow tables varies from 15 to 80, while the number of clients is 60 and the switches CPU capacity is 100 Gbps.

Figure 7 indicates a descending trend of β value for different flow tables. Small number of flow tables (<65) lead to a high value of β and load unbalancing. As the number of flow tables increases (>65), the trend has declined to indicate the achievement of load balancing. However, this trend stops and β reaches a constant value when the number of flow changes between 70 to 80. Thus, more number of flow tables (>70) cannot provide better load balancing. This value can be defined as a threshold value of flow tables (NFT_{Thr}) providing the best load balancing. As expected, the 2Q2L topology provides the best load balancing specially after 70 flow tables. This improvement is achieved due to the combined features of two other topologies (e.g. 1Q2L and 2Q1L)

Figure 7: β (in Logarithmic scale) for different number of flow tables, 60 clients and switches CPU capacity = 100 Gbps

C. CPU capacity variation

The switches with high CPU processing capacity have a positive effect on network performance. However, their high cost is a limitation. Therefore, it seems necessary to select a switch with a suitable CPU processing capacity. In this section, we evaluate the load balancing of different proposed topologies when CPU processing capacity varies. In this scenario, the number of flow tables and the number of clients are 216 and 50, respectively. The Switch CPU capacity varies between 1 and 17 Mbps. As shown in Figure 8, β value decreases as the processing capacity of the switches increases. When the processing capacity reaches 16Mbps, the reduction rate stops and β reaches a constant value. The simulation results indicate that increasing the CPU processing capacity more than a threshold value (here is 16 Mbps) does not affect the β value and thus the load balancing. Therefore, the selection of switches with proper CPU processing capacity can provide the required network performance in terms of load balancing at a reasonable cost. According to the results obtained from Figure 7, the 2Q1L topology performs better than other topologies when the CPU capacity changes.

To perform an accurate analysis of the 2Q1L topology, we set up two more simulations. The first one is the β value for variation of CPU capacity and network size (Figure 9). For small network size (Clients=20), satisfying load balancing is achieved with low CPU capacity (4 Mbps). As the larger network size, the required CPU capacity increases (16 Mbps for 50 clients). In the second simulation, the throughput of the border switch ($S=S_1+S_2$) is measured with a variation of CPU capacity and network size (Figure 9). The simulation results of Figure 10 demonstrates that saturation throughput depends on two variables: CPU capacity and network size. The required CPU capacity providing the saturation throughput is directly proportional to the network size. The coefficient of this relation

is almost 0.105. Thus, the saturation throughput (S_S) can be obtained as follows:

$$S_S = 0.21\ N_C$$

Where N_C denotes the number of clients. Moreover, the threshold of CPU capacity (CC_{Thr}) leading to the saturation throughput can be obtained as follows:

$$CC_{Thr} = 1 + 0.3\ N_C$$

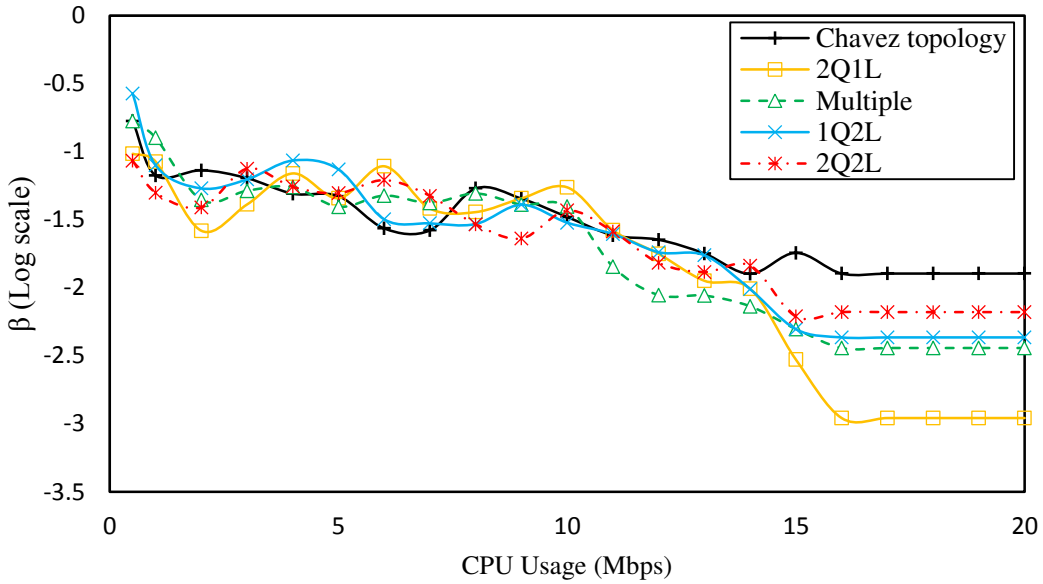


Figure 8: β (in Logarithmic scale) with CPU capacity variation, 50 Clients and number of flow tables=216

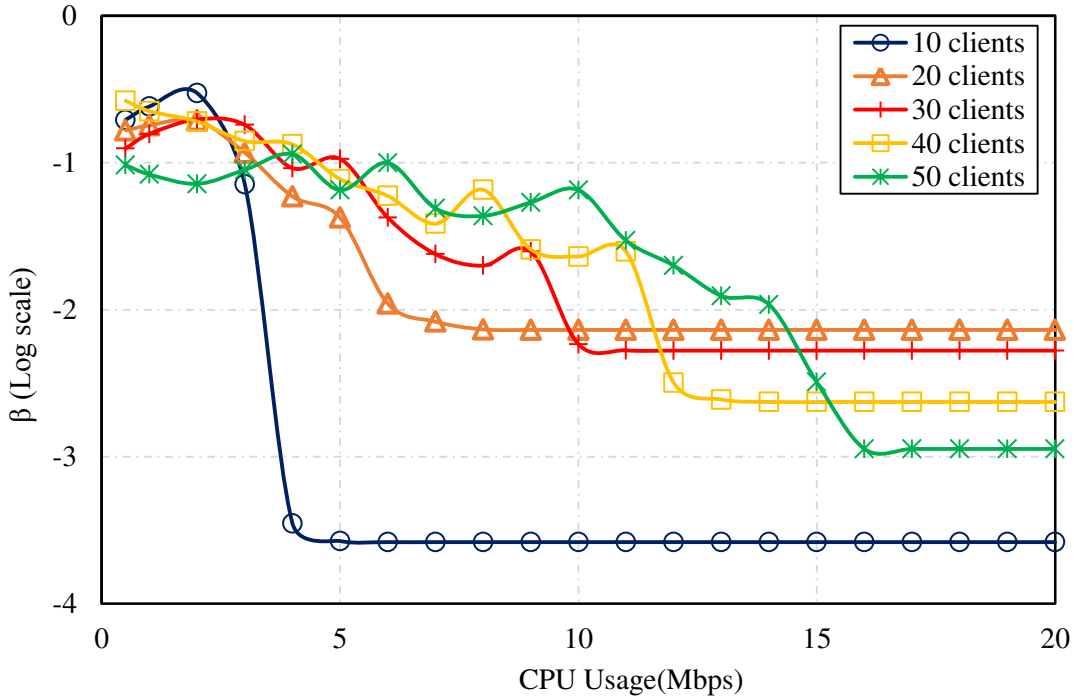


Figure 9: β (in Logarithmic scale) in the 2Q1L topology versus CPU capacity variation and network size variation when number of flow tables=216

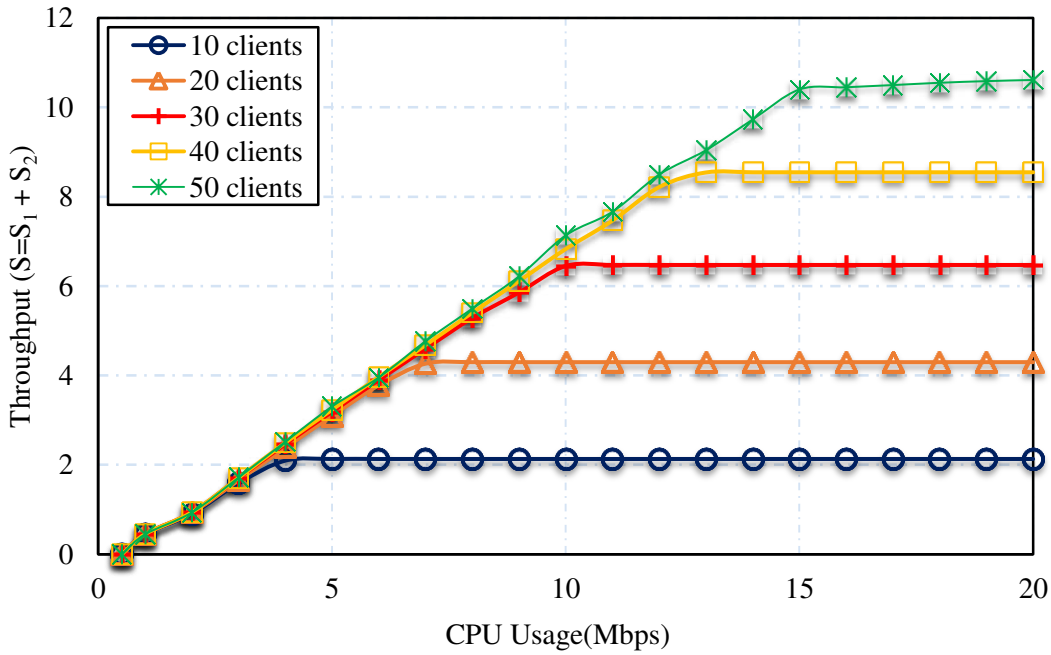


Figure 10: Throughput of the 2Q1L topology versus CPU capacity variation and network size variation when number of flow tables=216

4. Conclusions

Software-Defined Networking (SDN) improves traffic distribution and Quality of Service (QoS) in large scale networks. However, load balancing is one of the main challenges associated with efficient resource management and utilization. In this paper, four topologies have been proposed: 2Q1L, Multiple, 1Q2L, and 2Q2L. The proposed topologies are simulated using the OFSwitch13 module of the ns-3 network simulator in different scenarios. All proposed topologies outperform the Chavez topology [21] in terms of load balancing. However, 2Q1L topology presents the best performance in different simulation scenarios. Moreover, the simulation results determine the threshold values of effective parameters leading to the best load balancing on the servers. These variables are the number of flow tables and switch CPU processing capacity. Therefore, the proper selection of the effective parameters can provide satisfying performance with the minimum cost.

As the future research lines, other aspects of SDN challenges with a combination of the load balancing can pose a multi-objective optimization problem. These aspects can be throughput improvement, energy efficiency, and QoS provisioning.

Abbreviations

SDN: Software Defined Networking; *CPP*: Controller Placement Problem; *OFSwitch13*: Switch with protocol of OpenFlow 1.3; *OS3E*:Open Science, Scholarship and Services Exchange; *QoS*: Quality of Service; *2Q1L*: 2 QoS and 1 Learning; *1Q2L*: 1 QoS and 2 Learning; *2Q2L*: 2 QoS and 1 Learning; *CDF*: Cumulative Distribution Function; *TCAM*: ternary content-addressable memory; NFT_{Thr}: threshold value of flow tables; S_S: saturation throughput ; N_C: The number of clients; CC_{Thr}: the threshold of CPU capacity.

Acknowledgments

Not applicable

Authors' contributions

This manuscript is a contribution that originates from the master thesis of H.N. R. S and S.M.F.I contributed to the topologies of Load Balancing in Software-Defined Networking using Controller Placement. All authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE,* vol. 103, no. 1, pp. 14-76, 2014.

[2] D. Meyer, "The software-defined-networking research group," *IEEE Internet Computing,* no. 6, pp. 84-87, 2013.

[3] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Computer Networks,* vol. 81, pp. 79-95, 2015.

[4] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE communications surveys & tutorials,* vol. 16, no. 4, pp. 1955-1980, 2014.

[5] H. Uppal and D. Brandon, "OpenFlow based load balancing," *CSE561: Networking Project Report, University of Washington,* 2010.

[6] M .Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust sdn control plane for transactional network updates," in *2015 IEEE conference on computer communications (INFOCOM)*, 2015: IEEE, pp. 190-198.

[7] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement

problem in software defined networking: A survey," *IEEE Network,* vol. 31, no. 5, pp. 21-27, 2017.

[8] L. Li and Q. Xu, "Load balancing researches in SDN: A survey," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2017: IEEE, pp. 403-408.

[9] Guo, Z., Su, M., Xu, Y., Duan, Z., Wang, L., Hui, S., & Chao, H. J. (2014). Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Computer Networks*, *68*, 95-109.

[10] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 121-126.

[11] M. Qilin and S. Weikang, "A load balancing method based on SDN," in *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, 2015: IEEE, pp. 18-21.

[12] M. Qilin and S. Weikang, "A load balancing method based on SDN," in *Measuring Technology and Mechatronics Automation (ICMTMA), 2015 Seventh International Conference on*, 2015: IEEE, pp. 18-21.

[13] N. Handigol, M. Flajslik, S. Seetharaman, N. McKeown, and R. Johari, "Aster* x: Load-balancing as a network primitive," in *9th GENI Engineering Conference (Plenary)*, 2010, pp. 1.2-

[14] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-n-Serve: Load-balancing web traffic using OpenFlow," *ACM Sigcomm Demo,* vol. 4, no. 5, p. 6, 2009.

[15] Handigol, Nikhil, Srini Seetharaman, Mario Flajslik, Aaron Gember, Nick McKeown, Guru Parulkar, Aditya Akella et al. "Aster* x: Load-balancing web traffic over wide-area networks." In *GENI Engineering Conf.*, vol. 9. 2010.

[16] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-Based Server Load Balancing Gone Wild," *Hot-ICE,* vol. 11, pp. 12-12, 2011.

[17] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access,* vol. 7, pp. 24290-24307, 2019.

[18] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012: ACM, pp. 7-12.

[19] McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner.., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, no. 2, pp. 69-74, 2008.

[20] D. Gajjar, H. Kotak, and H. Joshi, "Round Robin Load Balancer using Software Defined Networking (SDN)," *Capstone Team Research Project, University of Colorado Boulder,* 2016.

[21] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "Ofswitch13: Enhancing ns-3 with openflow 1.3 support," in *Proceedings of the Workshop on ns-3*, 2016, pp. 33-40.

Figures

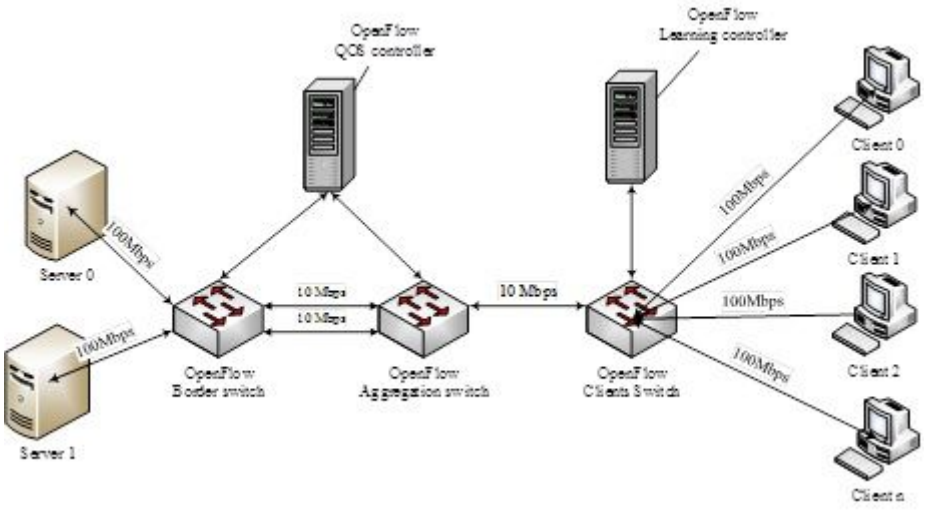


Figure 1

Chavez topology [19]

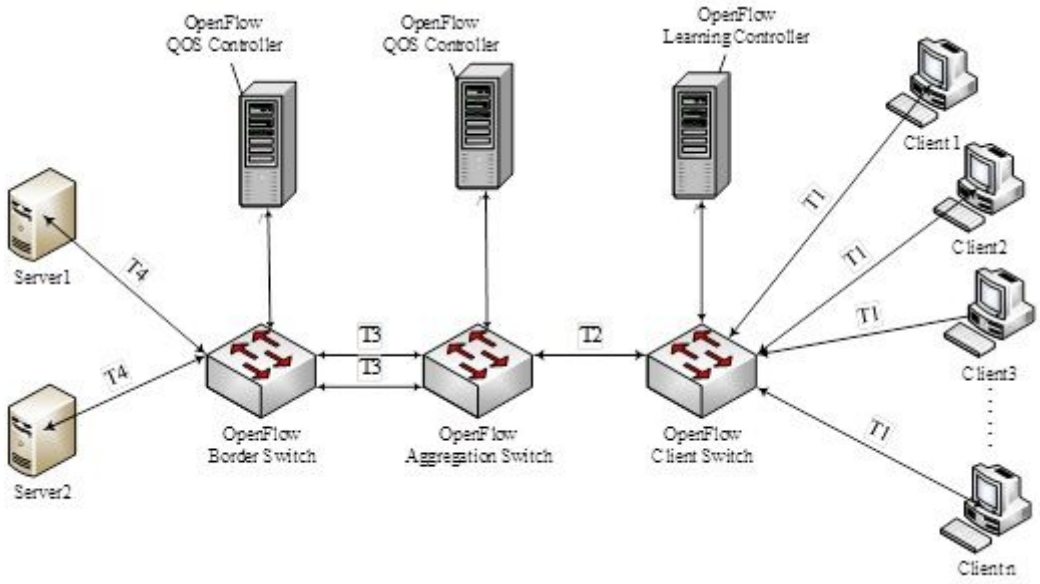


Figure 2

2Q1L topology

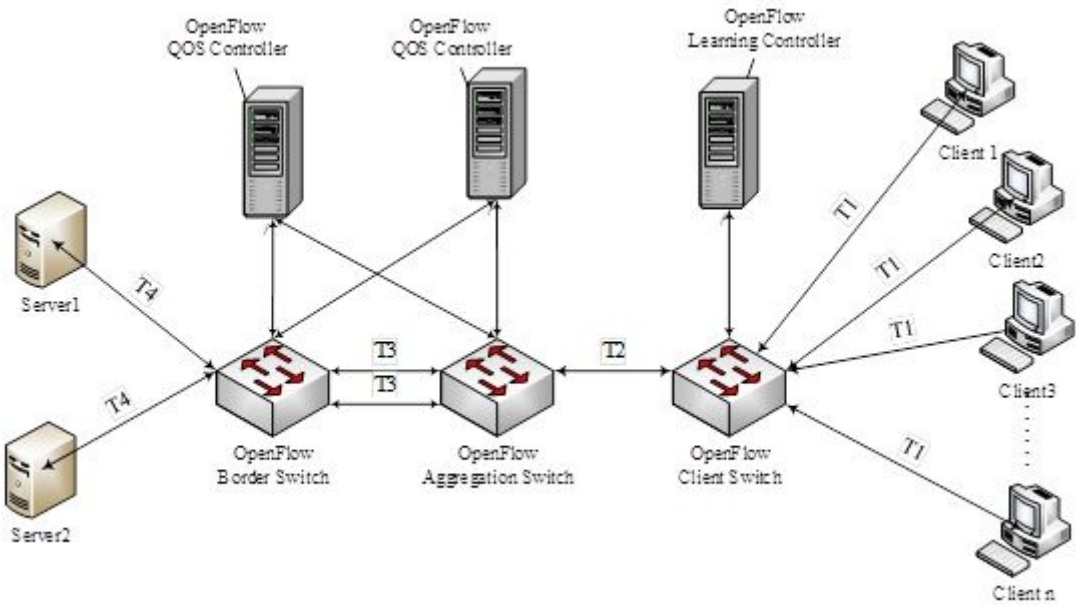


Figure 3

Multiple controller topology

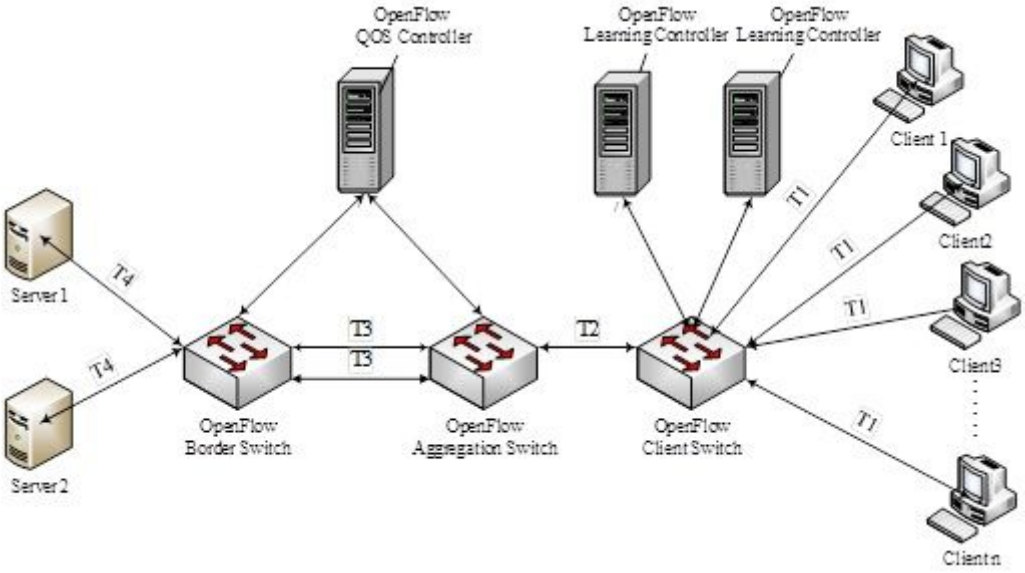


Figure 4

1Q2L topology

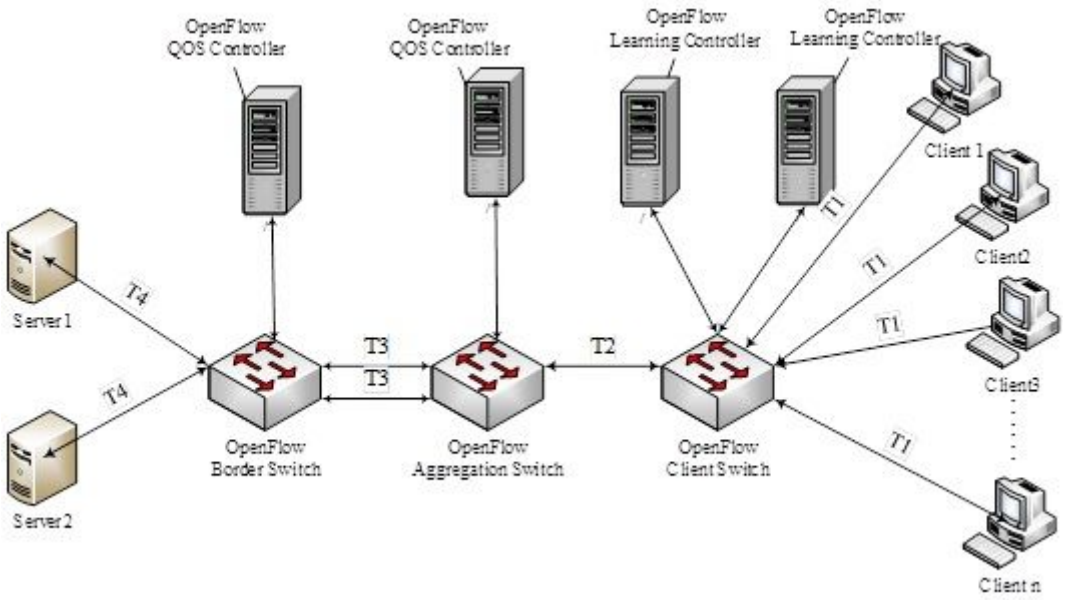


Figure 5

2Q2L topology

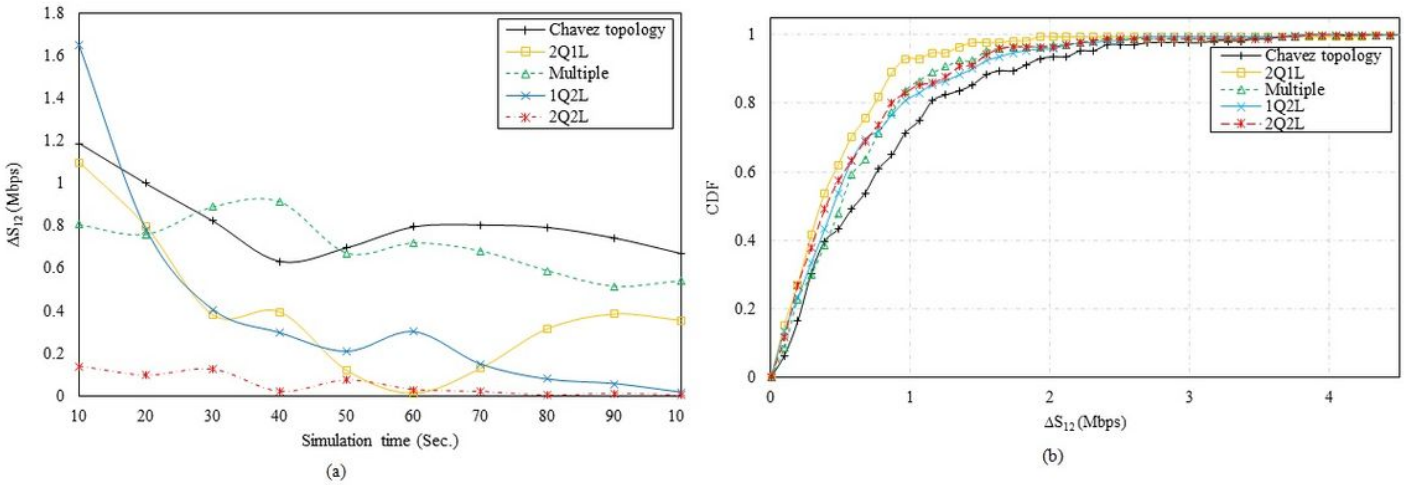


Figure 6

(a). The ΔS12 value for Number of Clients=20, CPU capacity = 100 Gbps and the number of flow tables=216 , (b).Cumulative Distribution Function (CDF) of ΔS1

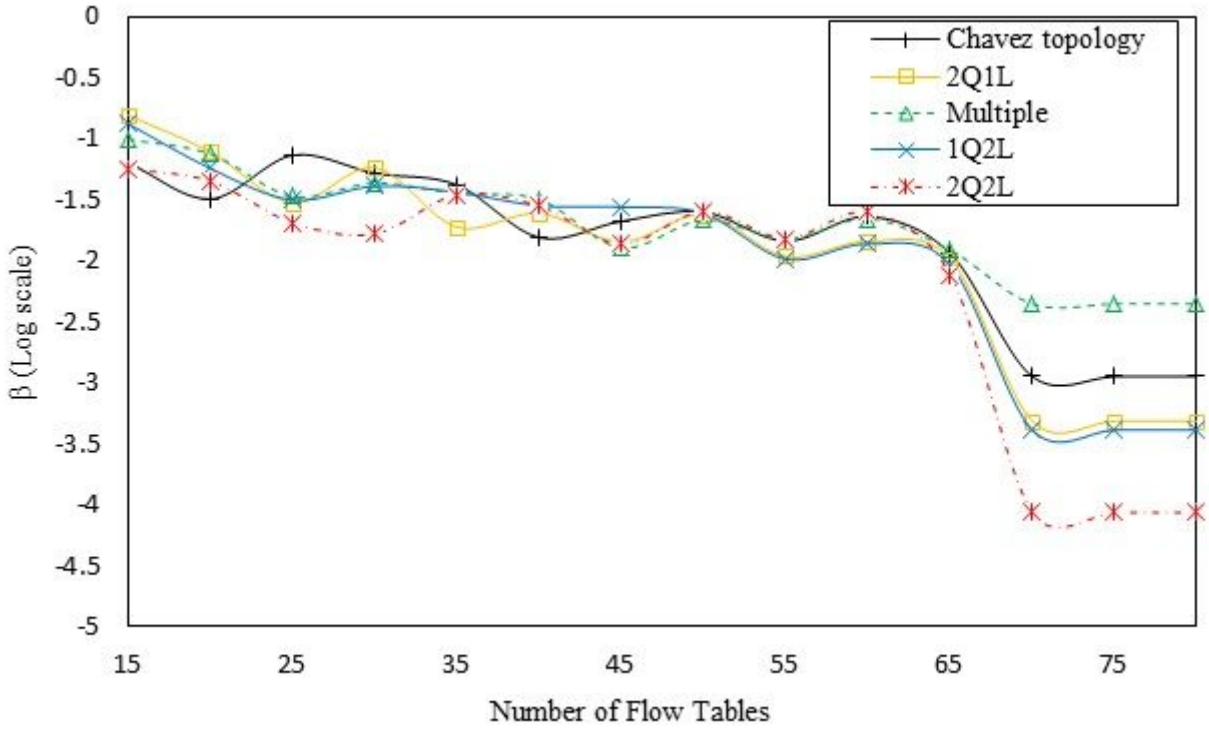


Figure 7

β (in Logarithmic scale) for different number of flow tables, 60 clients and switches CPU capacity = 100 Gbps

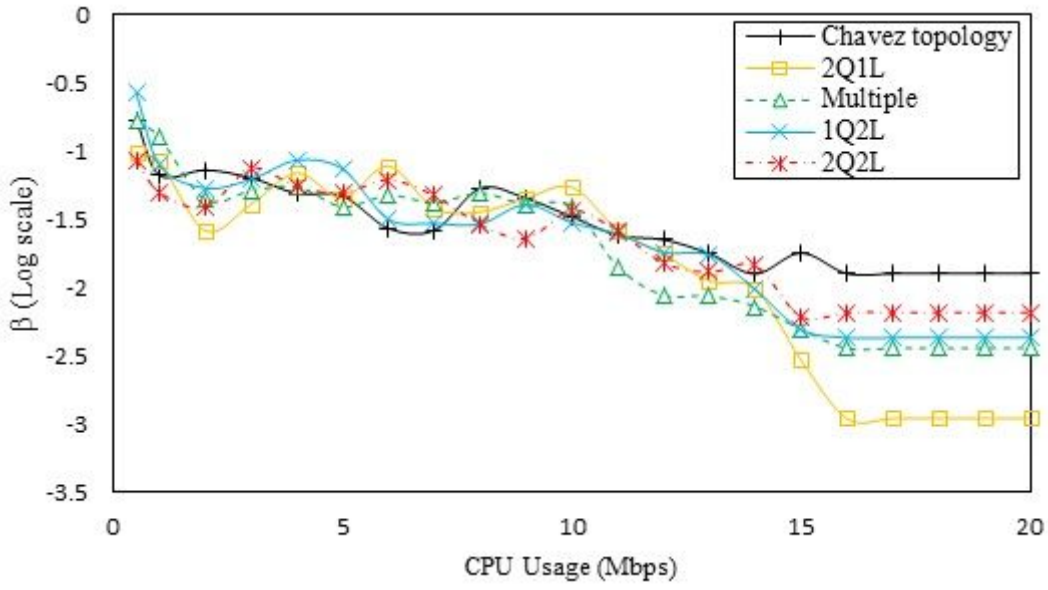


Figure 8

β (in Logarithmic scale) with CPU capacity variation, 50 Clients and number of flow tables=216

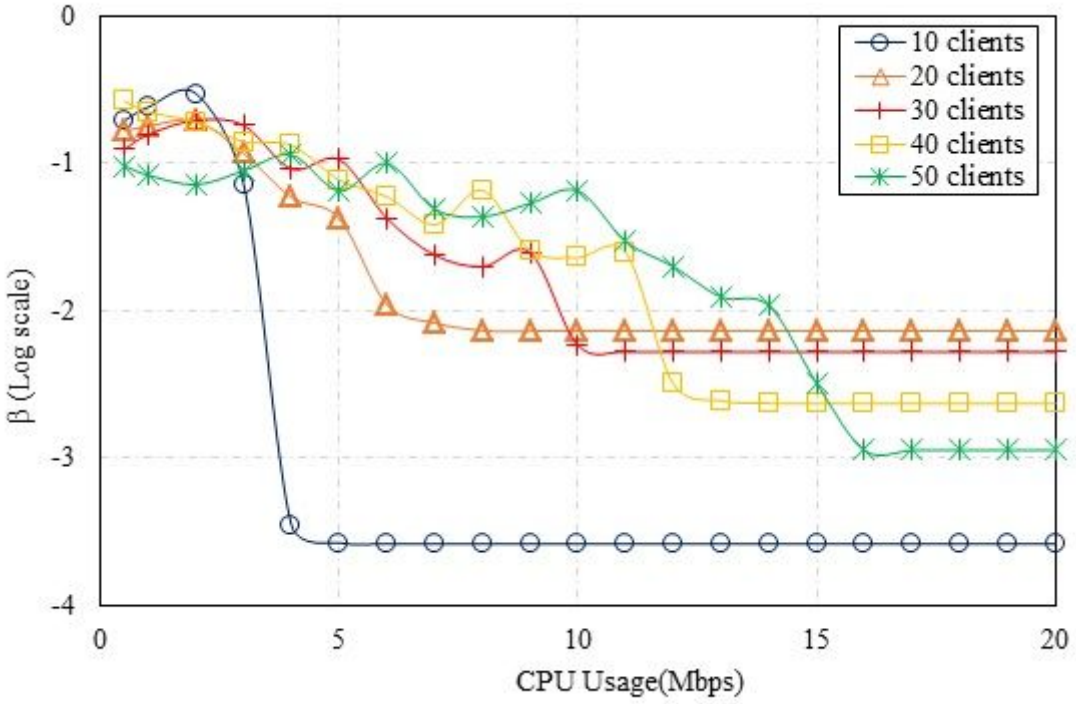


Figure 9

β (in Logarithmic scale) in the 2Q1L topology versus CPU capacity variation and network size variation when number of flow tables=216

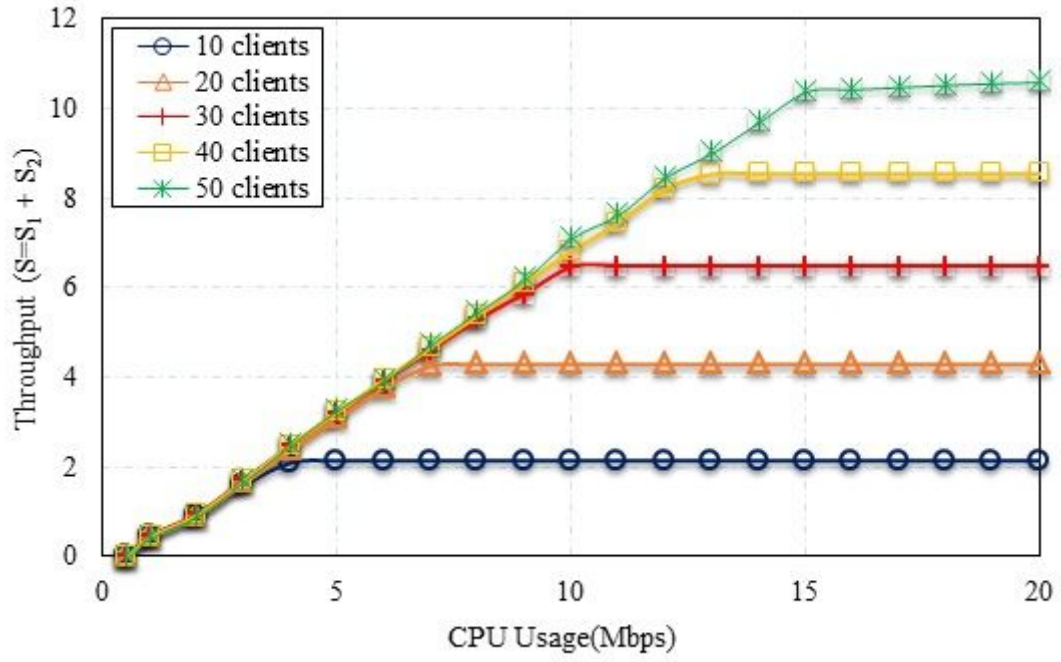


Figure 10

Throughput of the 2Q1L topology versus CPU capacity variation and network size variation when number of flow tables=216