Analyzing Current Load-Balancing Techniques in Software-Defined Networks for Increasing System Performance

Saugat Nepal

ISMT College, University of Sunderland, UK

Abstract

With the continued advancement of modern networks, application networking has emerged as an exciting new paradigm in today's networks. Even so, there are current issues with network performance and load redistributive taxation that must be addressed, and so many projects have been undertaken to tackle this problem. This study examines a portion of these papers on enhancing load-balancing network effectiveness in terms of link utilization, control-plane performance, or both, and compares the efficacy of proposed methods in various climates. Conclusions are presented at the end of this article, demonstrating the limitations of our current knowledge and which solution has the greatest potential for resolving the issue in most environments in contrast to others.

1. Introduction

Because of its capabilities, Software-defined networking is thought to be used in new promising paradigms for the future of the internet. Even so, as modern networks expand and users increase, one of the major challenges in this promising approach is an uneven distribution of traffic load in controllers, which leads to degraded system performance and denial of service to users for their services required (Priyadarsinia M. et.al. 2019).

Sun P. et al. (2020), for example, presented their dynamic controller workload trying to balance scheme MARVEL based on multi-agent relevance feedback instead of optimizing.

Numerous different methods were also introduced, as shown in the citations below. Ejaz S. et al. (2019) introduced traffic loaded trying to balance methodology in SDN-enabled networks by adding a secondary ISDN control system that is a reproduction of the initiation stage. The suggested fix improves the d performance of the system.

Cui J. et al. (2018) concentrated their efforts on SMCLBRT, a response time-based load-balancing strategy for multiple SDN controllers. Their assertion in this project is that they can solve the load-balancing problem in SDN by selecting an appropriate time threshold.

The purpose of this survey article will concentrate on critical analysis and evaluation of current load-balancing strategies used in software-defined networks to enhance the performance of the network. With a focus on research articles solving the difficulties in areas of link utilization on, control-plane load-balancing. However, the cited linked research offers useful answers. They do not fall under the category on which the main body of this essay is concentrated. In order to draw conclusions and make comparisons on their efficacy in enhancing network performance, this study will go over the techniques employed, experiments carried out, and their outcomes.

1. Assessment of Present Load-Balancing Techniques in SDN

The emphasis of this subsection will be on load-balancing techniques for enhancing SDN network performance.

1.1 Load-balancing for Website Quality

Attar S. et al. (2017) investigated creating an algorithm that periodically calculates link utilization to monitor network state. By rerouting a minimal number of flows, the suggested method was created to prevent congestion in SDN and lower network overhead. When link use exceeds the rate threshold, the controller anticipates

congestion on the link and determines how much load needs to be moved. By including these fluxes, the flows are moved to an uncongested path.

Attar S. et al reported experimental setup was based on using Mininet to design the testing topology. The controller platform selected by the authors to run the suggested algorithm is called ONOS. The experiment compared performance metrics for packet loss and throughput with and without a congestion-aware algorithm.

In Figures 1 and 2, the experimental findings are displayed. According to the reported outcomes, the algorithm's performance improved when congestion was present.

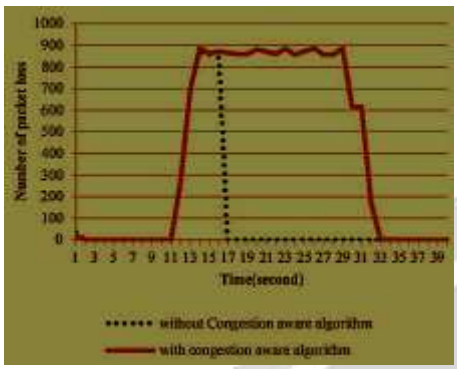

Figure 1 : Data packet loss rate, (Attarha S. et.al 2017)

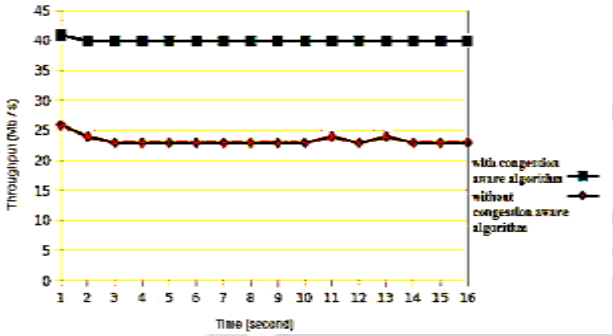

Figure 2: Efficiency with New Arrivals (Attarha S. et.al 2017)

Based on the findings, Attarha S. et al. (2017) came to the conclusion that the prosed algorithm is effective in resolving congestions by selecting a small number of flows and running quickly. Additionally, the results of the experiment indicate an improved performance in processing capacity and packet loss.

The fundamentals of the process were carefully thought out, with a focus on how to employ load-balancing in SDN to get improved results. The proposed research, however, does not account for additional elements like reaction times and delay. To adequately justify results, the method may have been contrasted with other load-balancing schemes. In order to confirm that the strategy is not underperforming in other areas, more data should be available.

In their research, Dewanto R. et al. (2018) described an SDN-based ECMP software that is intended to prevent network congestion problems. In advance determine the bandwidth capacity of each open path. When possible, the computer would route traffic along a non-overlapping path.

Rather than using max-min remainder capacity, Dijkstra's widest path algorithm was employed (MMRCS). Switches participating win the flow would update their flow tables once the path with the maximum available bandwidth was identified.

In the experimental setting created by Dewanto R. et. al. (2018), Ryu the controller platform was utilized to simulate a network topology based on a fat-tree architecture using a network simulation tool.

In comparison to standard ECMP during congestion time, the results reported in the work of Dewanto R. et. al. (2018) showed 8.7% lower packet, 14.21% higher throughput loss, and 92.27% lower delay. 3.46% more delay was given up in exchange for 0.19% less throughput in the same comparison during the non-congestion period. The suggested method demonstrated improvement over the round-robin by having a throughput that was 80.56 greater, an LSD value that was 75.2% lower, and 24.54% less packet loss.

Dewanto R. et al. (2018) came to the conclusion that their proposed strategy enhances the system's ability to balance based on the results. The trade-off highlighted during the non-congestion time will be the writers' main area of concentration.

A well-structured experimental setting was used to test the technique of the proposed scheme, and many scenarios were taken into account. The authors' suggested strategy has a minor shortcoming, according to a reliable evaluation. Consequently, their claims are well supported. To overcome the aforementioned constraint, additional effort is suggested.

To solve the problem of dispersing loads among controllers, Zhang S. et al. (2018) suggested online controller load balancing (OCLB). "Based on derived optimality and termination criteria of switch migration, an OCLB method is constructed." (2018) (Zhang S. et al.). In order to reduce the average controller response time, a system based on real-time request distribution has been developed. The technique is dependent on the controllers' load. There are two stages in the OCLB algorithm.

To obtain dynamic flow fluctuations, Zhang S. et al. (2018) ran simulations using Python that were wasted on data centers' work scenarios. A Fat-tree topology is utilized it is popular in data centers. On centers linked to edge gateways, controllers were installed. To test load management, online processing, and rate of convergence, three simulations were run. The scheme is contrasted with the greedy and optimal schemes. The Greedy approach, in which controllers frequently transfer the heaviest switch to the lightest operator, contrasts with Optimal's use of the Gurobi optimization.

According to the results, the suggested scheme is 5.79% better than the Greedy method and only 0.04% slower than the optimal reaction time on average. This result still falls within the proposed scheme's target of lowering connection speed. Additionally, as shown in Fig. 3, the results indicate that the online system can adapt to shifts in the timing of request distribution. Results in Fig. 4 showed that switch migrations increase linearly as more switches are added that exhibit extensibility features.

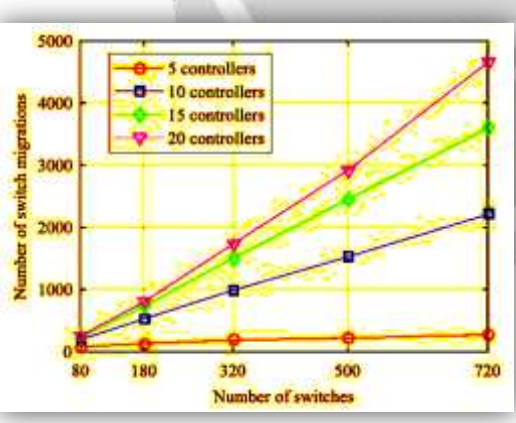

Figure 2: Shows the number of switch migrations required under various system settings to achieve load balancing. (2017) (Zhang S. et al.)

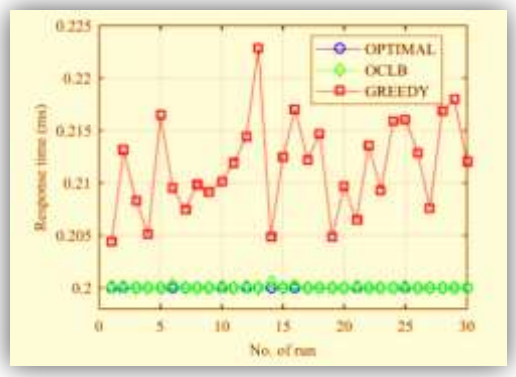

Figure 3 : shows the control plane's average response time.

Based on the obtained results, Zhang S. et al. (2018) came to the conclusion that their suggested method, OCLB, had achieved the desired outcome.

The methodology developed by Zhang S. et al. (2018) was rigorously validated utilizing topology in numerous circumstances to produce realistic results. The outcomes were contrasted with those of other approaches that had been examined in the same setting and under the same conditions. As a result, the findings are reliable and demonstrate that the suggested plan can enhance network performance.

Askar S. (2016) suggested an adaptable load-balancing system for SDN-based Fat-tree network topologies in data center networks. The intended scheme's goal was to flexibly balance DCN's load based on thresholds for capacity and packet loss triggering parameters. A centralized controller that had a complete understanding of the show's resources handled the load balancing. The programmer looks for other routes for applications with network delays or a high rate of packet loss.

A test-based approach Askar S. (2016) made the decision to use the Mininet tool to create a virtual environment and the MiniEdit GUI tool. Additionally, the studies — use of Open Flow switches, and the controller was set up using POX Controller as a platform for the suggested load-balancing mechanism. The experiment revolves around two situations that compare the effectiveness of task scheduling when connections between two hosts are already established and when a third host joins.

In the same virtualized environment, the proposed load-balancing scheme's outcomes were contrasted with those of the conventional scheme. For packet loss rate, the proposed adaptive load-balancing scheme improved performance by 81% in the first scenario and by 15% to 31% in the second. In the first situation, the algorithm increased throughput while in the second, it maintained overall throughput.

Based on the findings, Askar S. (2016) came to the conclusion that the suggested system has a significant advantage over a conventional load-balancing algorithm and is ready for application in such networks.

The methodology is sound, and sufficient information regarding the procedure is given. The experiment makes use of current technology, and the procedures are meticulously documented to ensure excellent repeatability. The outcomes demonstrated that the suggested approach is more effective than the established plan. All things considered, this research is conclusively justified, and no bias was added.

1.2 Load-balancing in the Control Plane

A global load-balancing cloud-balancing game theory and converging to Provides a well – provide equilibrium was proposed by Cimorelli F. et al. (2016). Designed to dynamically balance control traffic across SND cluster controllers. The proposed method has minimized latency and increased cluster bandwidth. That was said by the authors. This approach allows switches to select a handler without having to communicate with one another.

Cimorelli F. et al. (2016) used simulation with the MATLAB ® tool to test the suggested algorithm. On each switch, the algorithm was put into practice. Schemes practice late based on the latency and high of controllers relied on their reaction times. There have been ten models, each of which has been run ten times. The locations of the switches were chosen at random for each simulation. The proposed algorithm and the conventional switch-controller association strategy have been contrasted.

The proposed algorithm outperforms the nearest controller strategy, according to test findings. The outcomes are displayed in Fig. 5.

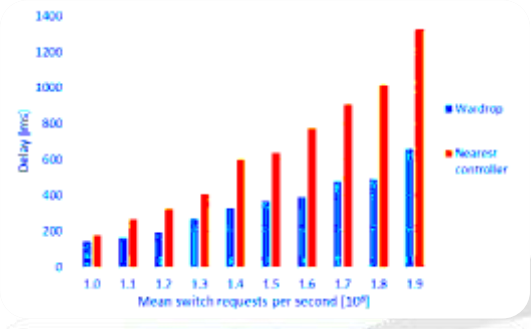

Figure 4 : Delays using the nearest controller strategy and the War drop strategy. (F. Cimorelli et al., 2016)

According on these findings, Cimorelli F. et al. (2016) concluded that the War drop technique can achieve its goals by overcoming the non-homogeneous geographical distribution of switches and controllers without explicit communication.

The outcomes of the proposed approach supported the claims because it was well thought out and rigorous. Using a trustworthy tool, a mathematical simulation was used to test the procedure, producing valid results. The research is reliable, but a lab test is suggested as a following step to thoroughly support the methodology.

An method for load balancing based on traffic patterns was described by Gasmelseed and Ramar (2018). This technique is hosted on distributed SDN controllers and was created to handle the TCP and UDP protocols. In order to distinguish between the TCP and UDP protocols, the algorithm checked and detected the header of incoming traffic flows. Fig. 6 depicts decision-making and detection. Following detection, traffic is distributed to a server farm via a dedicated control designed to handle the particular interface.

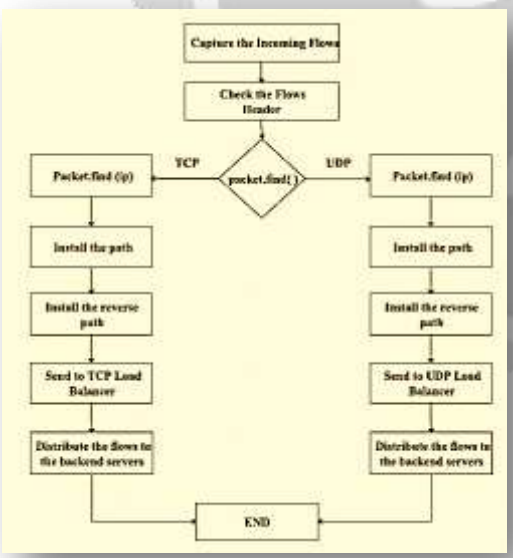

Figure 5: shows the flow of a traffic pattern-based load-balancing method (Gasmelseed H. and Ramar R. 2018)

Gasmelseed H. and Ramar R. (2018) used Mininet and Mini Edit to build an emulated SDN network based on distributed controller architecture in the experimental configuration. 10 backend servers were employed in the study, and the load balancing upload-balancing hosted on 4 controllers using the POX controller as a platform. A comparison of the method with arbitrary, round-robin, and balanced round-robin techniques was made.

According to the tests, incoming information by 11%, capacity climbed by 206%, packet loss was decreased by 86%, concurrency was reduced by 63%, and activity rate goes up by 258% when compared with the other techniques stated.

From the obtained results, Gasmelseed H. and Ramar R. (2018) came to the conclusion that their proposed algorithm uses distributed controller architecture to demonstrate its effectiveness in terms of the aforementioned metrics compared to other mentioned algorithms.

Regarding load-balancing in SDN, the approach put forth by Gasmelseed H. and Ramar R. (2018) offers a more straightforward solution. Unfortunately, this study just describes the experimental setup; it says nothing about the steps that were performed to execute the experiment. But besides this, the authors presented results that showed efficiency improvements in the specified metrics when compared to other algorithms using the same environment. Without this knowledge, the experiment cannot be repeated, and as a result, the results of this study cannot be supported.

1.3 Additional Load-Balancing Techniques

An operational load balancing on server response time was proposed by Zhong H. et al. in 2017. (LBBSRT). It also makes use of the SDN's flexibility advantage. The proposed method uses the controller's real-time measurements of each server's response to provide load balancing and an evenly distributed server load. "Server load capacity is directly reflected in server response time." (2017) Zhong H. et al.

According to Zhong H. et. At. Paperwork's (2017). Open vs. Switch developed the test-bed setup. Additionally, the controller for floodlights was chosen as an SDN controller. In order to provide online service and run WordPress to create a blog, three virtual machines with similar setups were deployed as web servers. To make an HTTP request to servers, two separate access frequencies were set up. One sends out requests continuously, while another does so every two seconds. 30 clients are being used by Topology to connect to the server and mitigate real-world scenarios.

The study's findings revealed that Round Robin, Random, and LBBSRT have average server response times of 1.236 seconds, 1.366 seconds, and 1.119 seconds, respectively.

Following a review of these findings, Zhong H. et al. (2017) came to the conclusion that their suggested system outperforms more widely used Round Robin and Random schemes in terms of load balancing.

The fresh proposed approach put forth by Zhong H. et al. (2017) was well justified because the researchers took traffic level variations into consideration to some extent. The findings' validity is demonstrated by a comparison to other load-balancing techniques. In conclusion, a strong case for it, and the suggested method enhances SDN functionality.

In their 2017 paper, Wang H. et al. mentioned load-balancing issues in SDN with links and controllers, and they presented a rounding-based method (RDMAR) to solve the issues. The goal was to simultaneously reduce controller reaction time and network use. Area-load boundaries and controller-load bounds are employed by the algorithm to make routing decisions.

Wang H. et al. (2017) used Mininet to establish a VL2 and Fat-tree topology for their testing environment. By contrasting the RDMAR algorithm with 5 more benchmarks, it was evaluated. Emergency accidents that disrupt traffic were also included. Testing was done to determine the effects of area load bound, running time performance, controller response time, and link load performance.

According to the findings, RDMAR can decrease the maximum link load in the two topologies by 37% and 17%, respectively. Comparing controller reaction times to other benchmarks, there was a 30% to 70% reduction. Compared to RDMAR-F, RDMAR lowered link-load by the 12%. Furthermore, RDMAR needs on about 0.19 seconds while NOX-MT needs roughly 0.6 seconds to decide on a route.

According to Wang H. et al. (2017), the outcomes demonstrated the effectiveness of the suggested method. The experiment demonstrated that, in contrast to existing methods, the algorithm substantially decreased controller reaction time while attaining a similar performance of link load balancing.

The suggested approach is important since it addresses connection issues as well as network control issues to improve load-balancing for SDN. A thorough testing strategy was utilized to utilize the suggested strategy with

other prevalent technologies. Running the experiment in two different topology situations supports the test results. The conclusion reached by the researcher is sound.

2. **Evaluation of Load-Balancing Methods for Enhanced SDN Performance**

In contrast, the Connection strategy suggested by Zhang S. et al. (2018) would function well in a wide network setting because it is extensible and relies on distributed controllers that share workloads to achieve this. The approach developed by Askar S. (2016) uses a single controller with a comprehensive network overview, which requires a lot of computational resources, rendering it more appropriate for small and medium systems.

The method of Cimorelli F. (2016), which makes use of numerous controllers and clusters, can theoretically achieve the best performance in big networks due to its capacity to balance control traffic between them. The network's capabilities are used more effectively the larger it gets. It is more extensible by core than the other algorithms presented in this work.

The suggested approach by Zhong H. et al. (2017) is best suited for data center centers because it relies on server response time. Therefore, this algorithm's full potential could not be utilized in networks without servers. The technique developed by Wang H. et al. (2017), in contrast, has a broader application and addresses both link and controller plane issues, making it more appropriate for use in broad networks than in data centers solutions

The analysis and comparison of several load-balancing strategies for SND have been conducted. The analysis of recent literature shows that numerous scholars developed numerous approaches to address the issue.

Along with Cimorelli F. (2016), Zhang S et al. (2016) provided a good solution for big networks with the ability to scale and maintain a competitive advantage.

The Askar S. (2016) solution significantly enhanced performance. The scaling solution, however, was lacking, and it was possible that the method would perform poorly in larger networks.

However, the method proposed by Zhong H. et al. (2017) was created to operate in server hosting networks. A solution that works with additional devices could be produced by combining one method with another.

3. Conclusion

The comparison of the works by Dewanto R. et. al. (2018) and Attarha S. et. al. (2017) revealed that while both groups' procedures are sound and reliable, only small- to medium-sized SDN networks will benefit from their use. An issue in a network with many data centers is the centers on a centralized.

The method developed by Wang H. et al. (2017) is the most appropriate of all the alternatives since it is extensible and provides answers for both connection and problems to improve network performance. Real-world application testing should be done on the most robust solution to demonstrate its efficacy.

References

Askar S, 2017, 'SDN-Based Load Balancing Scheme for Fat-Tree Data Center Networks', *Al-Nahrain Journal for Engineering Sciences (NJES)*, Vol. 20 No.5, pages 1047-1056

Attarha S., Hosseiny K. H., Mirjalily G., Mizanian K., 2017, 'A Load Balanced Congestion Aware Routing Mechanism for Software Defined Networks', *25th Iranian Conference on Electrical Engineering, pages* 2206-2210

Cimorelli F., Priscoli F. D., Pietrabissa A., Celsi L. R., Suraci V., Zuccaro L., 2016, 'A Distributed Load Balancing Algorithm for the Control Plane in Software Defined Networking', *24th Mediterranean Conference on Control and Automation (MED)*, pages 1033-1040

Cui J., Lu Q., Zhong H.,Tian M., and Liu L., 2018, 'A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time,' in *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1197-1206

Dewanto R., Munadi R., Negara R. M., 2018, 'Improved Load Balancing on Software Defined Network based Equal Cost Multipath Routing in Data Center Network', *Journal Infotel,* Vol 10, Iss 3, pages 157-162

Zhang S., Lan J., Sun P., and Jiang Y., 2018, 'Online Load Balancing for Distributed Control Plane in Software-Defined Data Center Network', *IEEE Access,* Vol. 6, pages 18184 – 181