Computers, Materials & Continua
DOI:10.32604/cmc.2022.017715
Article

Tech Science Press

Effectively Handling Network Congestion and Load Balancing in Software-Defined Networking

Shabir Ahmad[1], Faisal Jamil[2], Abid Ali[3], Ehtisham Khan[4], Muhammad Ibrahim[2] and Taeg Keun Whangbo[1,*]

[1]Department of I.T. Convergence Engineering, Gachon University, Sujeong-Gu, Seongnam-Si, 461-701, Gyeonggi-Do, Korea
[2]Department of Computer Engineering, Jeju National University, Jeju-Si, 63243, Jeju, Korea
[3]University of Engineering and Technology, Taxila, 47080, Pakistan
[4]The University of Haripur, Haripur, 22620, Pakistan
[*]Corresponding Author: Taeg Keun Whangbo. Email: tkwhangbo@gachon.ac.kr
Received: 08 February 2021; Accepted: 24 May 2021

Abstract: The concept of Software-Defined Networking (SDN) evolves to overcome the drawbacks of the traditional networks with Internet Protocol (I.P.) packets sending and packets handling. The SDN structure is one of the critical advantages of efficiently separating the data plane from the control plane to manage the network configurations and network management. Whenever there are multiple sending devices inside the SDN network, the OpenFlow switches are programmed to handle the limited number of requests for their interface. When the recommendations are exceeded from the specific threshold, the load on the switches also increases. This research article introduces a new approach named LBoBS to handle load balancing by adding the load balancing server to the SDN network. Besides, it is used to maximize SDN's reliability and efficiency. It also works in coordination with the controller to effectively handle the load balancing policies. The load balancing server is implemented to manage the switches load effectively. Results are evaluated on the NS-3 simulator for packet delivery, bandwidth utilization, latency control, and packet decision ratios on the OpenFlow switches. It has been found that the proposed method improved SDN's load balancing by 70% compared to the previous state-of-the-art methods.

Keywords: SDN; control plane; load balancing; decision tree; CPU utilization

1 Introduction

In recent years, multimedia technology has shown tremendous growth. This advancement has enabled high-quality video streaming and other applications; however, they suffer from heavy congestion and load on the network; thus, one specialized server must monitor the load balancing in the network. This balancer acts as the bridge between the network and server. The load balancer also monitors the server health based on the load balancing perspective not to configure

and implement different networking protocols [1]. The use of learning technologies is currently untilized in the network systems to make informed decisions about the load and its balancing and find optimal routes. However, such applications do not scale well, and scaling becomes cumbersome due to the non-convergence of optimization paths [2].

Moreover, the vast storage and high coupling between the logical structure and data transmission structure in the traditional network has become very complex in the conventional network structure and device control and its management on different scenarios [3,4]. To improve network performance and network control, the researchers at Stanford University U.K. presented the idea of a highly manageable and controllable network infrastructure called SDN. Ahmed et al. [5] state that SDN is an adaptable, cost-effective, effortless, and dynamic networking infrastructure for converting the traditional closed network infrastructure into an open SDN infrastructure. SDN's fundamental aim is to provide the interface to develop the software that can control and manage the network resources and traffic, along with possible modification in the traffic flow and control [6]. It has three primary layers; Infrastructure layer, also called Data Plane, Control Layer, also termed as Controller Plane; and Application layer, also known as Application Plane. OpenFlow switches are used to regulate the flow of the network data according to rules and protocol represented in a table called the flow table. The controller is considered the SDN brain; it collects the higher layer applications/software information. It maintains a flow table for allocation to OpenFlow switches through the OpenFlow protocol. The SDN controller also acquires network topology information and then provides a broad and open network view to the OpenFlow switches. Load balancer effectively distributes the network traffic to different paths based on the network congestion criteria. SDN architecture's primary goal is decoupling the data plane and control plane to enhance networking functionality.

In addition to the decoupling of the control plane and data plane, SDN offers many benefits. The virtualization of physical networks allows separation from the physical network. As a consequence, the physical network does not affect the corresponding logical network. Additionally, the open-source API in SDN allows a more customized and manageable network business. As users interact with only the upper layer, the user application layer provides a user interface for handling networks to meet their different needs. Finally, the separation of control plane and forward data plane is crucial for customers to manage their network management, innovations and flexibility. The centralized control provides the control and other administrative operations over the network, such as upgrades, business configuration speed. The generic structure of SDN is depicted in Fig. 1.

For load balancing and traffic engineering, centralized management is desired due to its faster convergence to the optimization objective and higher network performance. Since SDN works at the same network, the network switches can be managed and monitored by SDN central control. However, in a peak situation when the traffic increases on switches than its threshold, certain backup plans are required for graceful degradation. The increase in traffic is due to the number of requests generated by switches and nodes. End-to-end packet delay from a host to another host is upper bounded. Therefore, to tackle this challenge, we propose introducing a new module named LBoBS to dynamically offload the overly-congested node and maintain balance over the network.

The rest of the paper is generally organized as follows; Section 2 covers the related studies on Load balancing problems in SDN. Section 3 presents the design of the proposed solution, and Section 4 exhibits the simulation results. Finally, Section 5 concludes the paper and identifies future directions for this work.

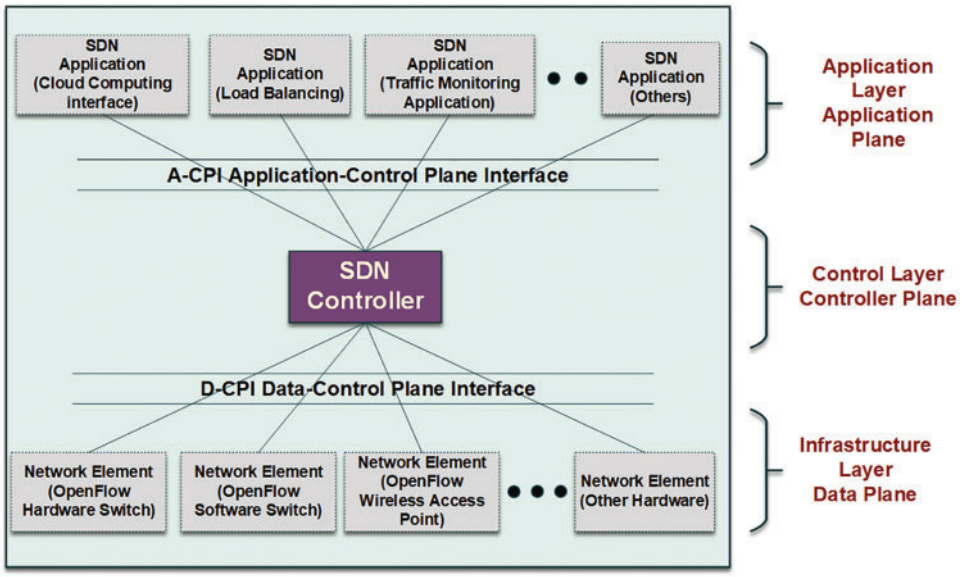

Figure 1: SDN architecture

2 Related Work

Various attempts have been made over the past few years to contribute towards load balancing challenges in contemporary SDN networks. The Equal-Cost Multipath approach [7] distributes the data load and flow to the hopes/switches without prior knowledge. William et al. [8] proposed the Valiant Load Balance approach that distributes all the traffic to different paths by picking the random next hope based on the random picking technique.

Handigol et al. [9], Wang et al. [10], and Wang et al. [11] proposed improved load balancing strategies and claimed that the controller is the critical component to handle the load balancing problem. The controller node monitors the response time from OpenFlow switches and updates the flow table to apply the load balancing technique specified by each system to balance SDN networks. However, one of the disadvantages of these strategies is they all are static in nature; thus, no real-time monitoring of traffic is available inside these strategies. Li et al. [12] proposed a novel load balancing technique based on a dynamic approach known as dynamic load balancing (DLB). The idea behind the DLB is to apply the greedy approach to pick up the next hope/switch, which transmits the minor data loads. DLB technique only decides the load on the next hope without determining the load on a global approach. In the global view of the transmission, this algorithm does not find the best path; hence, it does not achieve the system's best load balance effect.

Koushika et al. [13] proposed a new load balancing technique based on the Ant Colony Optimization approach. This technique finds the best path and the best server combinations for efficient path collection and optimization to collect the information from the network for link usage and calculate the delays in the links. However, this approach is a simple method to apply the network path information on a single criterion and thus does not scale well with future networks. Guo et al. [14] proposed that the controller complete a series of Real-time Least loaded Server selections (RLSs) for multiple domains to find the highly loaded table and direct the new flow to the least loaded server. It is also used to compute a path leading to the target server. One of the

problems with this approach is whenever a new flow enters into a domain; the RLS makes the forwarding decision for every new flow. However, this technique poses a problem such as a single controller bottleneck, poor scalability, reliability, and responsiveness.

In a nutshell, load balancing in the SDN has been extensively carried out in the recent past and different strategies are adapted to mitigate this hurdle. However, many strategies, despite allocating the best decision path, do not offer resource optimization when a load is adequately handled. SDN has been presented as an approach to bring high programmability to network components by decomposing a network's forwarding function into an efficient, fast path detection, and a programmable slow path. The extensibility is introduced through the latter, enabling new routing and forwarding approaches without replacing hardware components in the core network.

Load handling and balancing are significant issues in the SDN network, which causes network latency and slower response time. At times, packets may lose their path by searching the new switch for path forwarding and routing. Some approaches use SDN, virtualization [15–18], and contemporary methods employed in the Internet of things networks [19–22]. Nevertheless, a state-of-the-art load balancing mechanism is still considered at its infant stage for modern SDN networks. This research proposed a new technique for efficient forwarding of packets after facing their issues. We evaluate the parameters for performance, outlined in Tab. 1, mainly, Packet Decision Time on Server, Path Detection, Throughput, Bandwidth, and caching issues

Table 1: Load balancing techniques and their relations

Ref. paper	Methodology	Pros	Cons
Ali et al. [23]	Variance based load synchronization	1. Better performance 2. No packet loss 3. Overcome the sync overhead over controller	1. No evaluation of latency values 2. No energy consumption is evaluated
Jinke et al. [24]	HybridFlow Multicontroller load balancing approach	1. Reducing the Load on multi-controller. 2. Through the Load on super controller to others.	1. Complexity of calculations is very high. 2. Algorithm overhead and its the technique is not fit for throughput.
Nkosi et al. [25]	A dynamic load balancing based technique for cloud center (based on SDN approach)	1. Improve throughput. 2. Load does not cause an issue over an extended time.	1. Availability is low 2. Scalability is low
Yao et al. [26]	SDN-based dynamic load balancing technique between multiple mobilities	1. Improve the uplink and downlink traffic disruption	1. Bottleneck 2. Load detection is not evaluated. 3. Degree of load balancing is not considered. 4. Throughput is not considered. 5. Availability is low

(Continued)

Table 1: Continued

Ref. paper	Methodology	Pros	Cons
Yong et al. [27]	Lightweight load balancing technique	1. Improved throughput 2. High Efficiency. 3. More quick action when network traffic is changed dynamically.	1. Latency is not evaluated. 2. Complexity is High. 3. Energy consumption is not considered.
Raza et al. [28]	Load Balancing based on Server Response Time (LBBSRT)	1. Efficiency is high 2. Response time is minimum. 3. Maximum Availability. 4. Low cost	1. Bottleneck. 2. Availability is low 3. Scalability is low 4. Energy saving is not considered
Song et al. [29]	Load Balancing technique QoS (QALB)	1. Total network load in minimum. 2. Load balancing is improved. 3. Average OLR is reduced 4. QoS data rates are high	1. Scalability is low 2. Availability is low 3. Delay is high 4. Bottleneck
Zhong et al. [30]	Two Tier Dynamic load balancing approach	1. Wi-Fi re-association time is improved. 2. Wi-Fi load balancing improved approach	1. Bottleneck 2. Availability is low 3. Scalability is slow 4. QoS constraint not supported.
Rangisetti et al. [31]	Genetic-bases load balancing technique	1. Degree of Load balancing is high 2. Better Performance	1. Bottleneck 2. Availability is low 3. Scalability is low 4. Overhead is now evaluated 5. Computational time is high
Lin et al. [32]	Variance based load synchronization	1. Better performance 2. No packet loss 3. Overcome the sync overhead over controller	1. No evaluation of latency values 2. No energy consumption is evaluated
Chou et al. [33]	Load Balancing based on load informing strategy	1. Load Balancing Locally 2. Balance Load on each controller 3. Overcome the sync overhead over controller	1. No evaluation of fault tolerance 2. Imbalanced loaded balancing for external balancers. 3. No load balancing technique in multiple heterogeneous loads balancing frameworks.
Koushika et al. [34]	Dynamic Load Balancing technique	1. Worked on the load balancer. Flexibility. 2. Reduce network response. Time. 3. Improve the overall performance of the network and reduce delay. 4. Can adopt before the network failure or after a link failure	1. Delay is the off-network path finding. 2. Network delay method is adopted 3. Scalability is low 5. Computational time is high

3 Proposed Model

The model of our proposed load balancer in SDN is shown in Fig. 2. SDN controller take cares of the issues such as load balancing, security, topology, monitoring, loading, forwarding, etc., across the network. The addition of a load balancer in the SDN network supports intelligent decision-making where the Load Balancer controls every SDN switch's load. In the proposed technique, the proposed server is used to handle load balancing problems in the SDN. As the SDN controller is one of the servers responsible for managing all the switches, real-time load and path calculation control the load balancing problems. A controller is connected to the load balancer and switches. So, the controller periodically connects and transmits the Load balancing information to the load balancer regarding the load and distribution of incoming packets to different nodes.

Load control is the primary responsibility of the Load Balancer. Whenever the controller needs to process the load balancing scheme, the balancer returns the load balancing condition based on the calculated load path. The load balancer is directly in contact with the switches. Due to this, transmission overhead on a controller is reduced, and there is a direct connection between the load balancer and open flow switches. Balancer on one end relates to a controller and on the other end connects with the open flow switches.

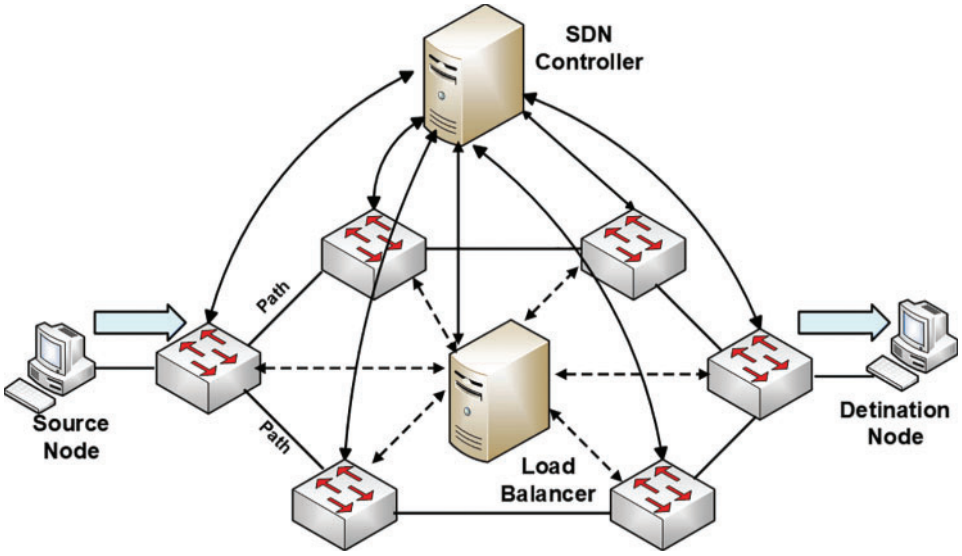

Figure 2: Proposed system's network architecture

Our proposed system model aims at efficient load balancing based on the SDN controller, Load Balancer, and open flow switches. Every SDN open flow switch contains a flow table for traffic flow information. This information is continuously updated with the consensus of load balancer and SDN controller.

The working of the proposed model is depicted in Fig. 3

a) Suppose a new data flow arrives, the open flow switch receives the information and matches the information with its internal flow information. If the information matches with the header of the flow table, i.e., if information persists, the data is transmitted according to the action fields.

If data is not found in flow tables, the SDN switch will send the packet header information to the load balancer and SDN controller.

b) Load balancer relates to the controller and open flow switches in direct connection. It takes and forwards commands to both SDN network devices for efficient flow information design.

c) The SDN controller is responsible for deciding the best transmission path in collaboration with the load balancer.

d) As the load balancer directly connects with the controller and switches, based on information from the controller and open flow switches, the SDN controller chooses one least loaded path; other paths that are not part of load balancing send this information to the controller.

e) The load balancer controller's information creates new flow tables with updated path information and least loaded path information and transmits this information to open-flow switches for transmission.

f) Controller and load balancer collaborate periodic information based on the open flow switch information.

g) Finally, the SDN controller creates a single path or multiple path information based on load balancer information.

As shown in Fig. 3, the direct link between the load balancer and switches and load balancer and controller effectively reduces transmission between controller and Load balancer and ultimately among switches and controller.

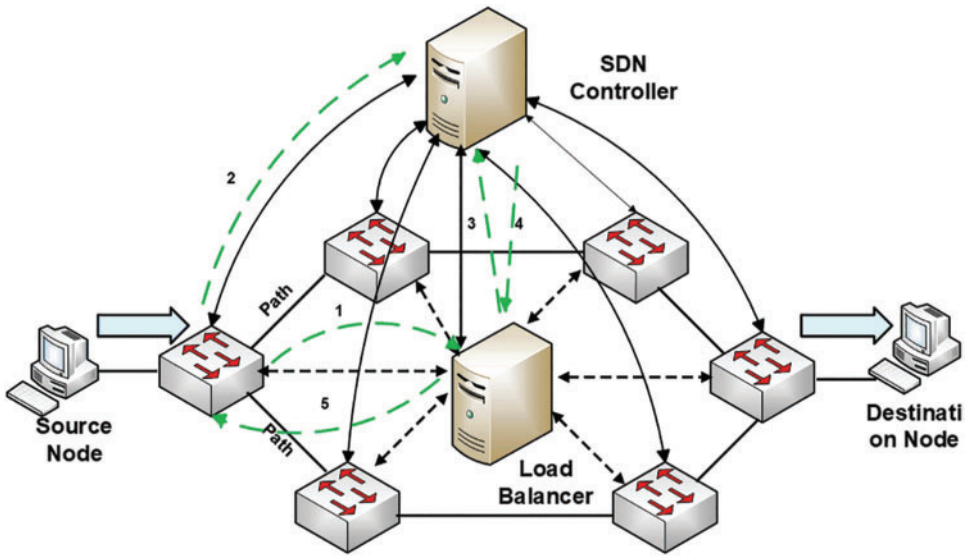

Figure 3: Control data/packets procedure plans

h) Suppose a new data flow arrives inside the SDN domain. The open flow switch receives the information match the information with its internal flow information regarding SDN network traffic. If the info matched the header information in the flow table and persisted in the flow table, the data is transmitted according to the action fields. If data does not find in flow tables, the SDN switch will send the packet header information to the load balancer and SDN controller.

i) Load balancer relates to the controller and open flow switches in direct connection. It takes and forwards commands to both SDN network devices for efficient flow information design.

j) The SDN controller is responsible for deciding the best transmission path in collaboration with the load balancer.

k) As load balancer directly connects with controller and switches, based on information from the controller and open flow switches, the SDN controller chooses one least loaded path; other paths that are not part of load balancing send this information to the controller.

l) The load balancer controller's information creates new flow tables with updated path information and least loaded path information and transmits this information to open-flow switches for transmission.

m) Controller and load balancer collaborate periodic information based on the open-flow switch information.

n) Finally, the SDN controller creates a single path or multiple path information based on load balancer information.

o) In the proposed model, the direct link between load balancer and switches and load balancer and controller effectively reduces transmission between controller and load balancer and ultimately switches.

In Fig. 4, we have evaluated the flow model of our approach in which the load balancing is considered at very high rates by using a load balancer for the effective mechanism.

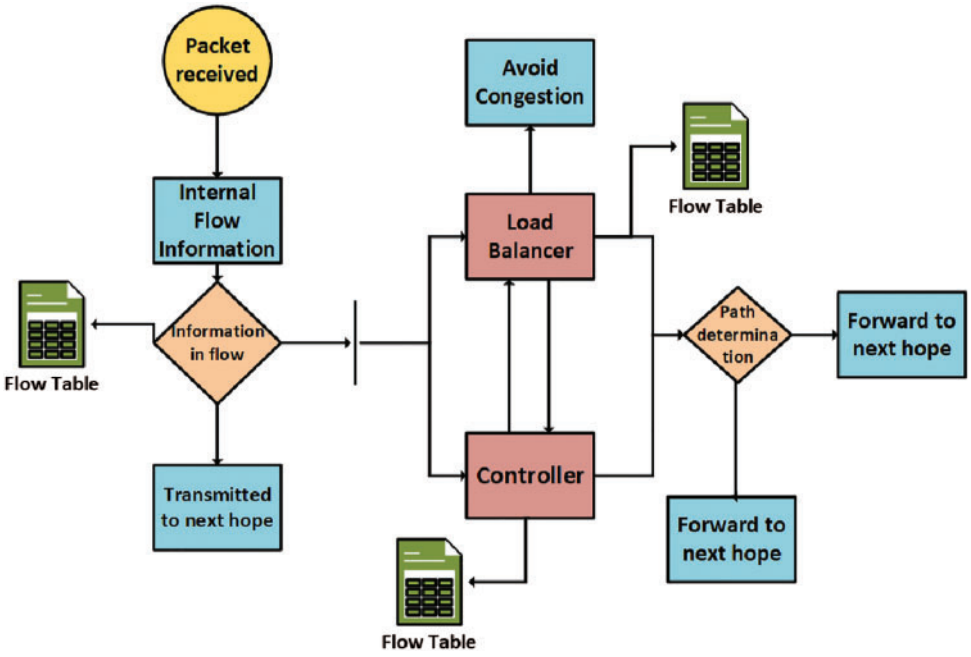

Figure 4: Flow of proposed model

The flow model contains the required information like path detection, load balancing, and path establishment. In this regard, we have evaluated the information and instruction flow model

diagram that shows an actual flow of instructions and data that passed from multiple filter and decision functions for the effective rates of methodology, as shown in Fig. 5.

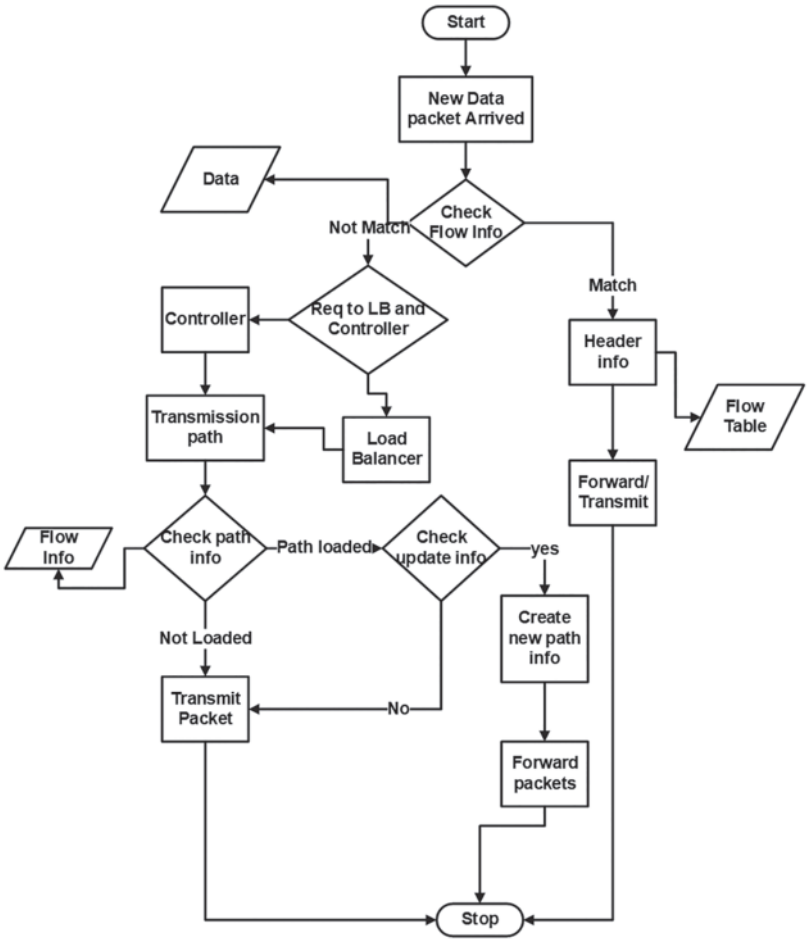

Figure 5: Data flow inside the proposed methodology

3.1 OpenFlow Switch and Controller Data Flow

The controller in SDN exchanges information with the SDN switch to forward the packets to the destination node and afterward delivers all of the concerned packets to the correct destination. The status information about each node's load is exchanged, tracked the loaded node, and triggers the algorithm to balance the load on the SDN network and forward packets-based information based on the information in the SDN flow table controller. Fig. 6 explores all of the data. The incomings packets in the SDN control and out port show the outgoing packets to the best and suitable path.

3.2 OpenFlow Switch and Controller with Load Balancing Server

To overcome the load balancing strategies on the controller, we introduce a new load balancing server. This load balancer directs the load balancing and network congestion in coordination with the SDN controller. The data packet rates may get high at some time so that switches and

controllers cannot control such a situation effectively, so that network congestion occurs in some places in the network. Fig. 7 shows the load balancer's coordination with the SDN controller to be divided with an adequate load balancing scheme.

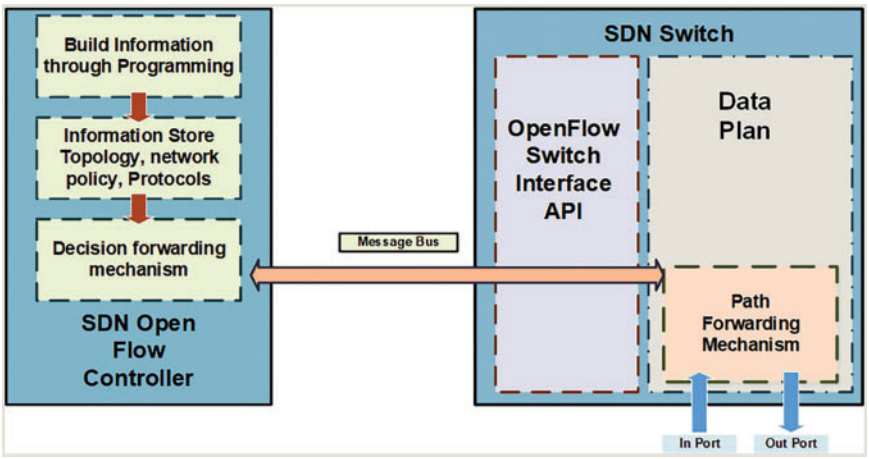

Figure 6: Controller and SDN switch information flow

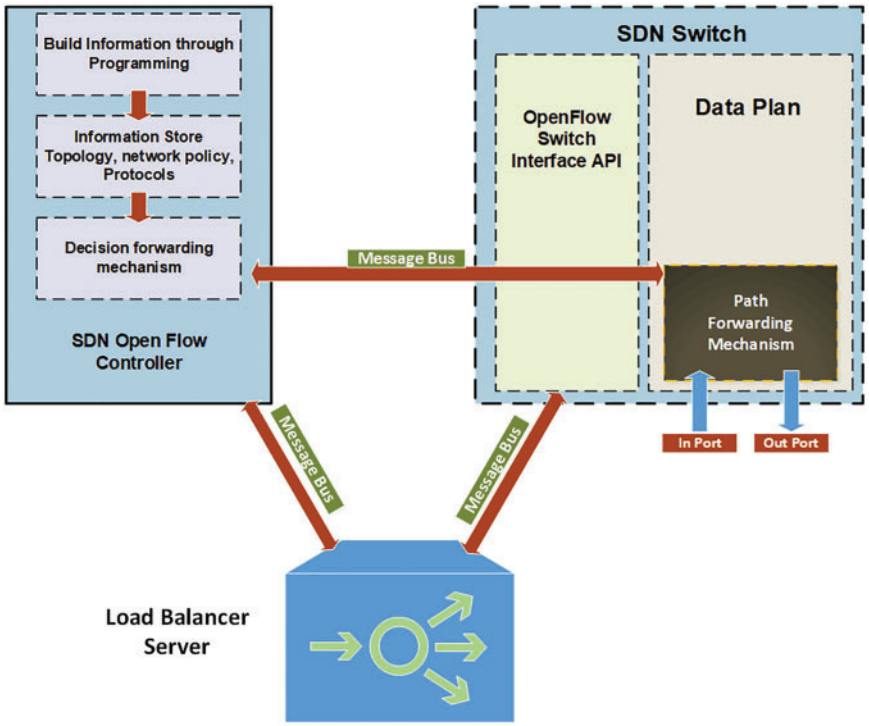

Figure 7: Load balancing scheme with a good load balancer in the SDN

4 Results and Discussions

This section describes the solution scenario for information-centric networks-based SDN load balancing and traffic congestion handling strategies. The simulation environment is set up for load balancing and congestion handling. Once the simulations are performed, the results are compared with existing methods. The basic parameters discussed are the packet decision time ratio, packet delivery, and bandwidth utilization.

4.1 Simulation Environment

We have deployed the NS-3 simulation environment [35,36] with the existing strategies to compare the results. The network environment is hosted on H.P. Elitebook 840 Pro with Intel Core i5-5200 CPU, 8 G.B. of RAM, 1 Tb SATA Hard Drive, Linux Operating System. Furthermore, we have deployed SDN Controller and Load Balancer Server and simulated traffic for a particular number of hosts and SDN switches. We have taken multiple senders and multiple receivers with several SDN switches and one SDN controller and load balancing server for balancing the load of the network. The load balancing server directly connects with the controller and SDN switches in our approach, so the load balancing load is divided.

Tab. 2 summarizes the symbols and notation used throughout this section. The notations, along with their brief explanation, are listed below.

Table 2: Notations

Notation	*Explanation*
$\wr$	Transmission path
P_k	Packet of request
Ns	Number of switches in the network
S_{witch}^{SDN}	SDN switch
L.B.	Load balancer
C_{SDn}	SDN controller
H_{info}	Header info
$\sum_1^n Li$	Links between nodes in SDN network
F_{info}	Flow information
F_T	Flow table
N_{path}	New forward path

Algorithm 1 exhibits the pseudocode for load handling policy. Symbols and notations used in it are already defined in Tab. 2. There should be no network congestion that will occur in this case. In this simulation environment, we have set up all the nodes, switches, controllers, and load balancers in the existing environment to produce the results. Connection setup can be made using wired and wireless connections. Some of the sending/source devices send the packets' devices, and some are receiving devices for the connection environment.

The complete working mechanism is described in Fig. 5, where all nodes are illustrated with relevant scenarios. We have compared achieved results with the relevant state of the art methods

shown in Tab. 3. The obtained results showed that our technique improved the load balancing and handling SDN network congestion quite effectively.

Algorithm 1: Load handling policy

Input: New Node, data packets
Output: Forwards packets on an efficient path.
Procedure:

a. Start
b. $req_a^{(New)} \rightarrow$ N
c. if ($f_{inf} \rightarrow$ match)
$H_{info} \rightarrow F_T$
$P_k \rightarrow$ frwrd ()
else if ($C_{SDN} \rightarrow req_a^{(New)}$ && LB $\rightarrow req_a^{(New)}$)
$C_{SDN} \Leftarrow \wr_{path}$
L.B. $\Leftarrow \wr_{path}$
path (C_{SDN}, LB)
else if ($\wr_{path} \Leftarrow F_{inf}$ (F_{table}))
$P_k \Leftarrow$ transmit ()
end if
end if
else
$N_{path} \Leftarrow$ path$\left(C_{SDn}, LB, S_{witch}^{(SDN)}\right)$
Packet_frwrd ()
d. end if
e. end

Table 3: Results comparison methodologies

Methodology	Parameter	Citation
Clustering and W/O-Clustering	Network latency	[37]
Advanced nearest neighbor load balancing (ANNLB)	Bandwidth utilization	[38]
Packet decision time ratio	Packet decision time ratio of a switch	[39]
Graph-based SDN switch	Packets delivery	[40,41]
EPE	Packets delivery ration	[42,43]

4.2 Results

The results are compared with the existing techniques shown in Tab. 3. Each testing policy is further evaluated with other methodologies for assessing its effectiveness. In the proposed method, the network topology matters for selecting packets, and we have used the SDN picketing set up for the effective results maintenance and results in a generation. We have applied the SDN protocols for the simulation environment and tested them to perform several tests. At the start of the simulation, we have forwarded fewer packets. Over time, we have increased the number of packets, data delivery ratio, and several SDN switches and consistently recorded all of the results. The best results are taken from the environment and compared with the strategies listed in Tab. 3.

4.2.1 Network Latency

Fig. 8a elucidates the network latency results in load balancing and SDN network congestion handling parameters. The x-axis represents the switches, whereas the y-axis depicts the latency in ms. Even for a massive number of switches of 25 or more, the recorded latency is 41000 ms for the proposed solution, and on the same number of switches, the other methods perform worse than the proposed method. Similarly, on increasing the number of switches to 150, the latency of the proposed method is slightly increased to 45900 ms, but in W/S-Clustering and Clustering techniques, the latency is more than 4650 ms. These results imply that the performance of the proposed system is better than W/S-Clustering and Clustering techniques in terms of latency. Over a keen look, it is found that the proposed technique reduces the network latency of load-balancing by 3.54%.

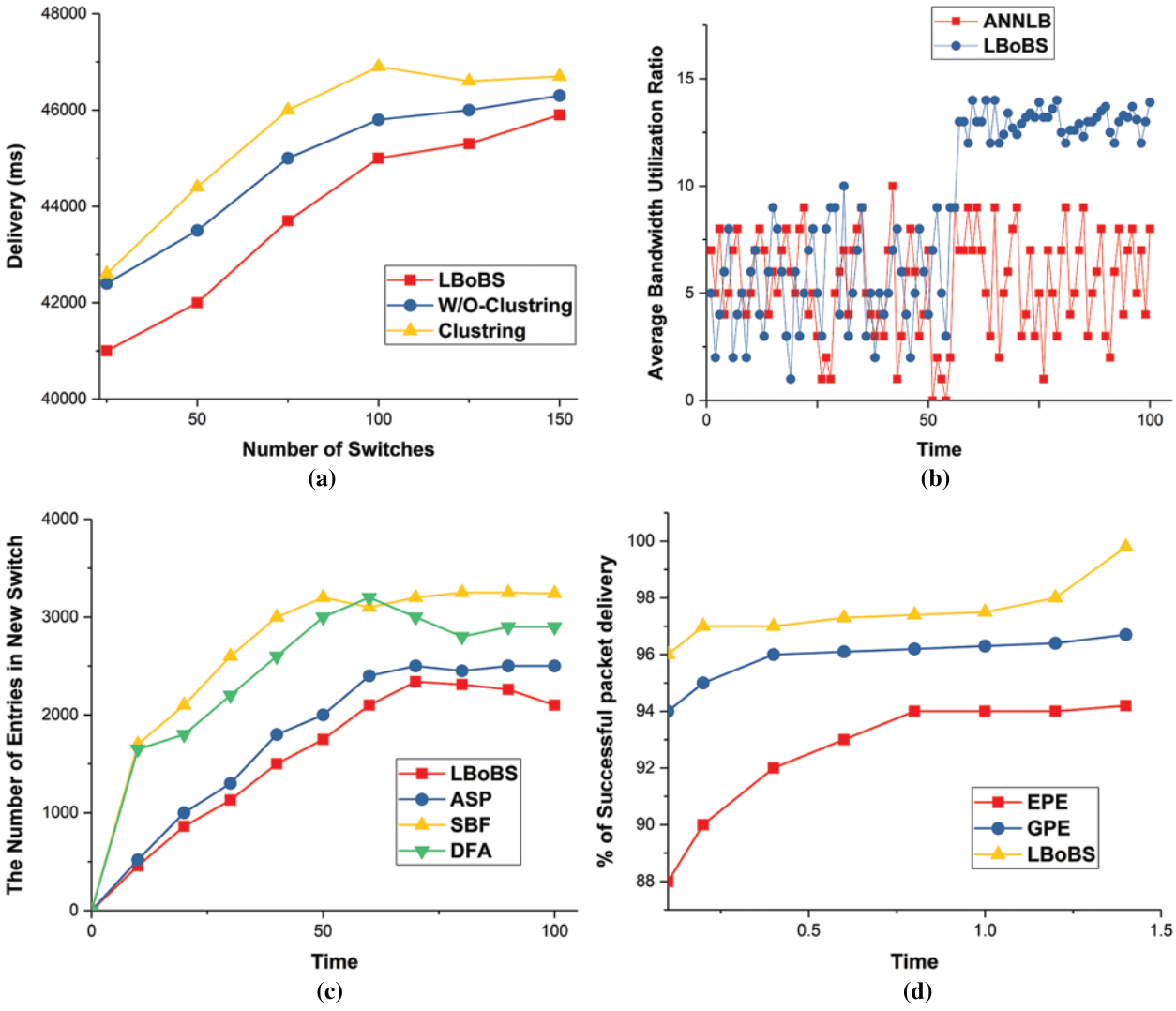

Figure 8: Performance evaluation. (a) Network latency (b) Bandwidth utilization (c) Packet detection ratio for switch (d) Packet delivery

4.2.2 Bandwidth Utilization

Fig. 8b exhibits the bandwidth utilization of the proposed method in comparison with other approaches to assess the performance of the proposed method. Results indicate the selection of best and most free path for incoming requests based on load balancing policy. ANNLB performs relatively similarly when the congestion is not high. However, for more congested routes, it tends to lose its effectiveness. In the proposed method, the bandwidth utilization is on a higher side and even achieve desirable results for highly congested paths. The proposed method not only chooses the best path to reduce latency but also improves bandwidth utilization. This bandwidth utilization increases the popular post-probability of congestion in the SDN network.

4.2.3 Packet Delivery Value

Fig. 8c shows the data packet flow entries inside the OpenFlow switch compared to the other three techniques. With just the simulation's start, we have set the level-0 for all of the entries. We have duplicate flow entries in all methods in the first second. Every second, all flow entries increase rapidly, but in the LBoBS technique, the increase is steady because the network packets are distributed, and load is balanced accordingly. With every second, the growth increases with time; thus, the increasing growth is slow compared to other techniques. After reaching the threshold values, the load balancing techniques show better results than different approaches.

4.2.4 Packet Delivery Value

Fig. 8d portrays the performance results of the LBoBs in terms of the packet transmission rate. From the graph, it is evident that it is more effectively deliver more packets compared to its counterparts methods. LBoBS detects the congestion and load on the network early with the load balancing server's help and applies the load balancing algorithm to identify the packets' new path effectively. The proposed policy is thus steady for the regular transmission of packets due to the high data rate. In comparison, other methods are not compatible due to the lower number of packet delivery.

5 Conclusion

This article investigated load balancing in the SDN-based networks where multiple servers are added, and numerous domains maintain them. Load balancing causes delay in the packet delivery, and at times packets may lose their way due to heavy load on some of the paths. Early path collision detection and decision on path analysis are the core contributions of this work. Load balancing and network congestion handling are ideal methods to handle the network load after implementing the load balancing server in coordination with the SDN controller and SDN OpenFlow switches. The load balancer continuously monitors the load on the SDN network. It directs the controller to change the packets' path to an alternate route in an undesired congestion situation until the load balancing issue resolves. It is an intelligent approach and is highly effective due to numerous reasons. Simulation results showed that the proposed LBoBS contributes significantly to load balancing through the handling of latency, bandwidth utilization, detection time ratio, and packet delivery ratio under different environments and scenarios. LBoBS also provides network congestion handling capabilities. In the future, we can extend this work by providing the load balancing and congestion over the ICN-based SDN, NDN-based SDN, and CCN-based SDN approaches.

Funding Statement: This research was supported by a Grant (21RERP-B090228-08) from Residential Environment Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

Conflicts of Interest: The authors declare that they have no interest in reporting regarding the present study.

References

[1] Q. Mao and S. Weikang, "A load balancing method based on SDN," in *Seventh Int. Conf. on Measuring Technology and Mechatronics Automation*, Nanchang, China, vol. 43, pp. 18–21, 2015.

[2] V. Srivastava and S. Pandey, "Machine intelligence approach: To solve load balancing problem with high quality of service performance for multi-controller-based Software Defined Network," *Sustainable Computing: Informatics and Systems*, vol. 30, no. 20, pp. 100511, 2021.

[3] X. C. Chui and Y. X. Bin, "Research on load balance method in SDN," *International Journal of Grid and Distributed Computing*, vol. 9, no. 1, pp. 25–36, 2016.

[4] N.O. Foundation, "Software-defined networking (SDN) definition," 2020. [Online]. Available: https://www.opennetworking.org/sdnresources/sdn-definition.

[5] A. Ahmed, T. A. Fong, A. Gani, U. Garba, S. Khan *et al.,* "Distributed controller clustering in software defined networks," *PLOS ONE*, vol. 12, no. 4, pp. 174715–174734, 2017.

[6] J. Christopher, O. Lyndon, S. Karthik and S. Vishnu, "Emerging transport SDN architecture and use cases," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 116–121, 2016.

[7] T. C. Wing and L. K. Yeung, "Traffic distribution over equal-cost-multi-paths," in *IEEE Int. Conf. on Communications*, Paris, France, vol. 2, pp. 1207–1211, 2004.

[8] D. William and B. Towles, *Principles and Practices of Interconnection Networks*, 2004. [Online]. Available: https://www.elsevier.com/books/principles-and-practices-of-interconnection-networks/dally/978-0-12-200751-4.

[9] N. Handigol, S. Seetharaman, K. Flajslik, N. McKeown, R. Johariet *et al.,* "Plug-n-serve: Load-balancing web traffic using OpenFlow," *ACM Sigcomm Demo*, vol. 4, pp. 6–7, 2009.

[10] R. Wang, D. Butnariu and J. Rexford, "OpenFlow-based server load balancing gone wild," in *Proc. of the 11th USENIX Conf. on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services USENIX Association*, Berkeley, CA, USA, vol. 6, pp. 12, 2011.

[11] Y. Hu, W. Wang, X. Gong, X. Que and X. Cheng, "BalanceFlow: Controller load balancing for OpenFlow networks," in *2nd Int. Conf. on Cloud Computing and Intelligent Systems*, Singapore, vol. 21, pp. 780–785, 2012.

[12] Y. Li and D. Pan, "OpenFlow based load balancing for fat-tree networks with multipath support," in *Proc. 12th IEEE Int. Conf. on Communications*, Budapest, Hungary, vol. 20, pp. 1–5, 2013.

[13] A. M. Koushika and S. T. Selvi, "Load balancing using software defined networking in cloud environment," in *Int. Conf. on Recent Trends in Information Technology*, Chennai, India, vol. 41, pp. 1–8, 2014.

[14] Z. Guo, M. Su, Y. Xu, Z. Duan, L. Wang *et al.,* "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Computer Networks*, vol. 68, no. 4, pp. 95–109, 2014.

[15] J. Ali, G. M. Lee, B. H. Roh, D. K. Ryu and G. Park, "Software-defined networking approaches for link failure recovery: A survey," *Sustainability*, vol. 12, no. 10, pp. 4255, 2020.

[16] S. Ahmad, A. Khudoyberdiev and D. Kim, "Towards the task-level optimal orchestration mechanism in multi-device multi-task architecture for mission-critical IoT applications," *IEEE Access*, vol. 7, no. 1, pp. 140922–140935, 2019.

[17] S. Ahmad, S. Malik, I. Ullah, D. H. Park, K. Kim *et al.,* "Towards the design of a formal verification and evaluation tool of real-time tasks scheduling of IoT applications," *Sustainability*, vol. 11, no. 1, pp. 204–226, 2019.

[18] J. Ali and B. H. Roh, "An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping," *IEEE Access*, vol. 8, pp. 88990–89006, 2020.

[19] S. Ahmad, L. Hang and D. Kim, "Design and implementation of cloud-centric configuration repository for DIY IoT applications," *Sensors*, vol. 18, no. 2, pp. 474–494, 2018.

[20] S. Ahmad, I. Hussain, M. Fayaz and D. Kim, "A distributed approach towards improved dissemination protocol for smooth handover in mediasense IoT platform," *Processes*, vol. 6, no. 5, pp. 46–61, 2018.

[21] J. Ali and B. H. Roh, "Quality of service improvement with optimal software-defined networking controller and control plane clustering," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 849–875, 2021.

[22] S. Ahmad and D. Kim, "A multi-device multi-tasks management and orchestration architecture for the design of enterprise IoT applications," *Future Generation Computer Systems*, vol. 106, pp. 482–500, 2020.

[23] J. Ali, B. H. Roh and S. Lee, "QoS improvement with an optimum controller selection for software-defined networks," *PLOS ONE*, vol. 14, no. 5, pp. e0217631, 2019.

[24] Y. Jinke, Y. Wang, K. Pei, S. Zhang and L. Jiacong, "A load balancing mechanism for multiple SDN controllers based on load informing strategy," in *2016 18th Asia-Pacific Network Operations and Management Symp.*, Kanazawa, Japan, vol. 14, pp. 1–4, 2016.

[25] C. Nkosi, A. Mpho, A. Lysko, Albert and S. Dlamini, "Multipath load balancing for SDN data plane," in *2018 Int. Conf. on Intelligent and Innovative Computing Applications*, Plaine Magnien, Mauritius, vol. 14, pp. 1–6, 2018.

[26] H. Yao, C. Qiu, C. Zhao and L. Shi, "A multicontroller load balancing approach in software-defined wireless networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, pp. 454159–454167, 2015.

[27] W. Yong, T. Xiaoling, H. Qian and K. Yuwen, "A dynamic load balancing method of cloud-center based on SDN," *China Communications*, vol. 13, no. 2, pp. 130–137, 2016.

[28] S. M. Raza, D. Park, Y. Park, K. Lee and H. Choo, "Dynamic load balancing of local mobility anchors in software defined networking based proxy mobile IPv6," in *Proc. of the 10th Int. Conf. on Ubiquitous Information Management and Communication*, Danang, Vietnam, vol. 24, pp. 106, 2016.

[29] P. Song, Y. Liu, T. Liu and D. Qian, "Flow stealer: Lightweight load balancing by stealing flows in distributed SDN controllers," *Science China Information Sciences*, vol. 60, no. 3, pp. 32202–32218, 2017.

[30] H. Zhong, Y. Fang and J. Cui, "LBBSRT: An efficient SDN load balancing scheme based on server response time," *Future Generation Computer Systems*, vol. 68, no. 1, pp. 183–190, 2017.

[31] A. K. Rangisetti and B. R. Tamma, "QoS aware load balance in software defined LTE networks," *Computer Communications*, vol. 97, no. 1, pp. 52–71, 2017.

[32] Y. D. Lin, C. C. Wang, Y. J. Lu, Y. C. Lai and H. C. Yang, "Two-tier dynamic load balancing in SDN-enabled Wi-Fi networks," *Wireless Networks*, vol. 24, no. 1, pp. 1–13, 2017.

[33] L. D. Chou, T. Y. Yang, Y. M. Hong, J. K. Hu and B. Jean, "A genetic-based load balancing algorithm in openflow network," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 1st ed., vol. 1. Berlin, Germany: Springer, pp. 411–417, 2014.

[34] A. M. Koushika and S. T. Selvi, "Load balancing using software defined networking in cloud environment," in *Int. Conf. on Recent Trends in Information Technology*, Chennai, India, vol. 16, pp. 1–8, 2014.

[35] J. Ivey, H. Yang, C. Zhang and G. Riley, "Comparing a scalable SDN simulation framework built on NS-3 and DCE with existing SDN simulators and emulators," in *Proc. of the 2016 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation*, Danang, Vietnam, vol. 17, pp. 153–164, 2016.

[36] N. N. Dao, J. Kim, M. Park and S. Cho, "Adaptive suspicious prevention for defending DoS attacks in SDN-based convergent networks," *PLOS ONE*, vol. 11, no. 8, pp. 160375–160399, 2016.

[37] S. Ahmad, S. Malik, I. Ullah, M. Fayaz, D. H. Park *et al.,* "An adaptive approach based on resource-awareness towards power-efficient real-time periodic task modeling on embedded IoT devices," *Processes*, vol. 6, no. 7, pp. 90–116, 2018.

[38] M. Hussain and N. Shah, "Automatic rule installation in case of policy change in software defined networks," *Telecommunication Systems*, vol. 68, no. 3, pp. 461–477, 2018.

[39] V. Srivastava, R. Pandey and S. J. S. C. I. Systems, "Machine intelligence approach: To solve load balancing problem with high quality of service performance for multi-controller-based software defined network," *Sustainable Computing: Informatics and Systems*, vol. 30, no. 11, pp. 100511, 2021.

[40] A. Abdelaziz, "Distributed controller clustering in software defined networks," *PLOS ONE*, vol. 12, no. 4, pp. e0174715, 2017.

[41] C. Chen-Xiao and X. Ya-Bin, "Research on load balance method in SDN," *International Journal of Grid and Distributed Computing*, vol. 9, no. 1, pp. 25–36, 2016.

[42] N.-N. Dao, J. Kim, M. Park and S. Cho, "Adaptive suspicious prevention for defending DoS attacks in SDN-based convergent networks," *PLOS ONE*, vol. 11, no. 8, pp. e0160375, 2016.

[43] M. Hussain, N. Shah and A. Tahir, "Graph-based policy change detection and implementation in SDN," *Electronics*, vol. 8, no. 10, pp. 1136, 2019.