IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 12, No. 10, October, 2025, pp. 3531-3544
Journal homepage: http://iieta.org/journals/mmep

Heterogeneous Traffic Management in SDN-Enabled Data Center Network Using Machine Learning-SPIKE Model

Sanarya Jamal Al-Azawee*, Nadia Adnan Shiltagh Al-Jamali

Department of Computer Engineering, University of Baghdad, Baghdad 10011, Iraq

Corresponding Author Email: gs22.sjalazawee@coeng.uobaghdad.edu.iq

https://doi.org/10.18280/mmep.121019

Received: 13 April 2025
Revised: 25 June 2025
Accepted: 4 July 2025
Available online: 31 October 2025

Keywords:
congestion, Dijkstra algorithm, Widest Dijkstra algorithm, elephant flows, mice flow, spike neural network, traffic flows

ABSTRACT

Software-Defined Networking (SDN) has evolved network management by detaching the control plane from the data forwarding plane, resulting in unparalleled flexibility and efficiency in network administration. However, the heterogeneity of traffic in SDN presents issues in achieving Quality of Service (QoS) demands and efficiently managing network resources. SDN traffic flows are often divided into elephant flows (EFs) and mice flows (MFs). EFs, which are distinguished by their huge packet sizes and long durations, account for a small amount of total traffic but require disproportionate network resources, thus causing congestion and delays for smaller MFs. MFs, on the other hand, have a short lifetime and are latency-sensitive, but they account for the vast bulk of traffic in data center networks. The incorrect use of network resources by EFs frequently disturbs the performance of MFs. To meet these issues, precise classification of network traffic has become crucial. This classification enables traffic-aware routing techniques. This paper offers a novel model for classifying SDN traffic into MF and EF using a spike neural network. Once identified, traffic is routed based on the classification results. For MF, the model uses the Dijkstra algorithm. For EF, the Widest Dijkstra algorithm is used. This model solves the difficulties of traffic heterogeneity in SDNs by integrating advanced classification techniques and strategic routing algorithms. It enables desirable resource allocation, eliminates congestion, and increases network performance and dependability. The models used have proven their efficiency by outperforming the traditional Software Defined Network and other algorithms in terms of: throughput by 60%, and 20%, bandwidth utilization by 5%, and 7%, packet loss by 50%, and latency by 60%, respectively.

1. INTRODUCTION

Most professional corporate organizations, regardless of size, believe that owning data centers (DCs) is essential for effective competition. However, the rapid increase and distribution of data centers complicate control and management operations. Operators can spend hours, days, or even weeks manually configuring networks for specific devices [1]. Large-scale data centers require a standardized approach to managing infrastructure. Controlling data centers can be challenging due to their dispersed locations and multiple infrastructure providers [2]. The Software-Defined Networking (SDN) overcomes the constraints of classical DC. SDN is the most recent advancement in networking technology. It makes network administration easier by separating the control and data planes. This improves network flexibility and efficiency. SDN architecture promotes simple configuration and troubleshooting methods [3]. The core notion of this design is decoupling, or the separation of the control and data planes [4]. The control plane is the system that handles traffic management, and hence the logic that determines where the packets that arrive should be transmitted [5]. This procedure results in the formation of a structure that holds all packet forwarding choices, such as a routing table. The data plane is the underlying mechanism that routes traffic to the next hop (next node) based on the structure created by the control plane. Currently, this separation results in basic network devices that just handle packet forwarding and hence only implement the data plane. The control plane is implemented as software, which is therefore totally programmable and referred to as a controller. The controller communicates with the devices via a southbound interface, with which it sends the information essential for the proper operation of the SDN switches, and outputs information in the form of an API using a northbound interface [6]. In today's systems, however, the controller employs both traditional artificial intelligence and neural network routing techniques. Flow routing is critical to enhancing network performance. The basic purpose of flow routing in a network is to get the data as rapidly as possible. Routing can increase network performance, which is an evident advantage [7, 8]. The traffic flow is diverse, with different arrival rates, durations, and sizes. The heterogeneity has an influence on both their Quality of Service (QoS) and network resource needs. They behave differently when traveling to their goal [9]. SDN data center traffic flows are often divided into two categories: elephant

flows (EFs) (large, long-lived) and mice flows (MFs) (small, short-lived). EFs, which are distinguished by their huge packet sizes and long durations, account for a small amount of total traffic but require disproportionate network resources. MFs, on the other hand, have a short lifetime and are latency-sensitive, but they account for the vast bulk of traffic in DCNs. The incorrect use of network resources by EFs frequently disturbs the performance of MFs, resulting in suboptimal resource use and QoS degradation. As a result, their actions produce congestion and delays in the vast majority of MF. However, MFs require high priority due to their delay sensitivity [9, 10]. In spite of the SDN architecture providing a global perspective and increased network programmability, allowing for flexible capabilities and effective QoS provisioning strategies [11]. Currently, most DCNs suffer from the exploitation of network resources by huge packets (elephant flow) that enter the network at any time, affecting MF. Poor network performance is mainly because of high congestion and unequal load due to inappropriate traffic distribution. Due to these problems, the classification of network traffic into MF and EF with their precise forecasting has become a necessity. Such a classification brings routing techniques of maximizing traffic-aware, which can allocate network resources depending upon flow characteristics, reducing packet congestion and delays. Exploiting the centralized control and programmability of SDN, flexible traffic management schemes can be constructed to moderate the conflicting requirements of different traffic types, ensuring optimal resource utilization and stable QoS.

This paper seeks to construct an SDN-based application employing supervised machine learning to meet specified goals.

- Create an SDN-based routing system that identifies flows. The system has two components that work with the controller: traffic classification and traffic routing.
- Identify flows as mice or elephants, using spiking neural networks (SNNs).
- Implement a flow routing algorithm that prioritizes shorter paths for mice and wider paths for elephants. As a result, the network becomes more balanced, leading to increased throughput.
- Optimize route cost calculation and ensure compliance with real-time data.

The proposed model addresses the challenges of traffic heterogeneity in SDNs via a combination of state-of-the-art classification approaches and optimal routing methods. It enables desirable resource allocation, eliminates congestion, and increases network performance and dependability.

2. RELATED WORK

Several types of research have been done in the area of SDNs. SDNs centralize routing control by transferring it from separate network parts to a single location. Furthermore, the synchronization and control capabilities of an SDN give all of the necessary information about the connectivity between hosts as well as the ability to make quick switching decisions. SDNs ensure high-level performance [9]. Network performance degradation is mostly caused by congestion and imbalanced load. Many researchers recommended strategies to alleviate network congestion and balance network load, while others advised enhancing traffic routing. Li et al. [12] suggested a dynamic multi-controller deployment strategy based on load balancing, which improves scalability and reliability in SDN systems. It turns flow requests into a queuing model and investigates traffic propagation latency and controller capacity. Zaher et al. [13] introduced Sieve, a revolutionary distributed SDN-based platform for flow scheduling that improves network performance and efficiency via intelligent flow management. Sieve initially organizes a subset of the flows based on available bandwidth, independent of class. Shirali-Shahreza and Ganjali [14] addressed the restricted flow table issue by delaying rule installation and hastening evictions to alleviate network congestion. Their approach anticipates TCP flow termination from RST/FIN packets to accelerate rule evictions while handling non-TCP flows to delay rule installation. It lowers the drop in traffic. Liu et al. [15] described a DRL-based intelligent routing solution for SD-DCN that improves network efficiency and performance. A DRL agent on an SDN controller learns from network data and makes adaptive routing decisions depending on its current state, such as bandwidth and cache, because the cache should impact routing decisions by removing duplicate traffic in the DCN. Kumar et al. [16] showed how machine learning algorithms may determine the least congested path for routing traffic in an SDN network. Their proposed route selection method creates a list of possible routes using network information from the SDN controller. Modi and Swain [17] proposed a routing approach for SD-DCN that employs RNN deep learning models (LSTM-RNN and BiLSTM-RNN). Their proposed model provided a routing path combination based on past traffic statistics, which improves the SDN controller's effectiveness. Mu et al. [18] suggested utilizing reinforcement learning to regulate SDN flow entry. Suggested an RL-based approach for minimizing control overhead by selecting appropriate parameters for rule insertion in Ternary Content Addressable Memory (TCAM) to overcome the constrained capacity of TCAM used in an OpenFlow-enabled switch. Improved network performance and traffic load. Lin et al. [19] proposed an efficient strategy for identifying elephant traffic in data centers. The OpenFlow controller can send individual or aggregate statistics messages to acquire statistical data from OpenFlow switches. The authors proposed a Hierarchical Statistics-pulling approach that uses aggregate statistical data to identify elephant movement in a smaller area. Using aggregate statistical data can significantly reduce bandwidth consumption during elephant flow detection. It submits an aggregate statistics request for the full flow space block. If a block's aggregate stats reply reaches the threshold, divide it into four equal-sized blocks. After four repetitions, it sends individual requests to all flows whose aggregate dependency exceeds the threshold, indicating an elephant flow. Aymaz and Çavdar [20] proposed an approach to detect EFs to improve routing efficiency. Their proposed model employed deep learning to identify EFs. Their classifier categorizes flow using a convolutional neural network (CNN) structure. As a result, certain EFs are routed. Yusuf et al. [21] suggested a method that worked with the SDN controller to establish paths for each traffic type. As a consequence, their proposed approach combines composite measurements with flow classification to detect congestion-prone flows and divert them along the most appropriate pathways to avoid congestion and traffic loss. Al-Saadi et al. [22] proposed an SDN application that uses network performance metrics, like flow time, number of packets, and average packet size, to select the optimal path for each flow type. K-means clustering is utilized to detect flows using these three metrics and classify them into

elephants and mice, then choose the path for each typical flow. Although relatively new and validated, supervised learning with SNNs has not been explored for traffic classification. It is a remarkable oversight, particularly because SNNs reduce energy consumption significantly, which is perfectly in line with the goals of the modern industry, where sustainability and energy efficiency have become paramount. SNNs are continuously proving themselves to be very efficient in many applications, including image classification [23] and robotics [24]. The SNNs draw inspiration from biological systems, enabling lower-power operation and improved performance, especially in scenarios involving real-time adaptive decision-making. Such properties are ideal and resolve the challenges of traffic heterogeneity in SDN, thereby making SNNs a promising option for addressing those problems. As previously stated, researchers have made various improvements to the SDN layers to address congestion and load balance by adding additional controllers, increasing the buffer size of the switches, performing routing operations based on specific parameters such as bandwidth and cache, and prioritizing the elephant flow over the MF. Hence, the SNNs opted to employ them for traffic classification in this work, as the benefits justify it. This is the first step to explore their potential for this field study. But the hope is to take advantage of their unique capability to build a complete but energy-efficient approach to tag network traffic as mice or elephants. Then, after classification, equal priority is given to both the mice and the elephant in the routing process by sending them in a way that suits the needs of both types and not encountering them on the same path. Such a novel approach not only fills existing traffic management research gaps but also helps to build sustainable, high-throughput SDN systems.

3. METHODOLOGY

This paper introduced two interrelated models, as shown in Figure 1, to characterize SDN-enabled traffic classification and routing applications. In the first model, a Spike neural network is applied to classify network traffic into both EFs and MFs. It is classified according to important traffic-generating parameters like flow time, packet size, and byte rate. SNNs are particularly applicable for implementing this model. The SNN can classify flows with high accuracy and in real time. The second model is related to traffic routing, following the classification technique. After that, the routing system selects the best path depending on classification results, consumes efficient network resources, and delivers QoS. On the other hand, the Dijkstra algorithm allows sending the MFs using the shortest path to affect the delivery time for the packets that require low latency. EF are generally bandwidth-intensive, which are divided over multiple paths instead of a single path using the Widest Dijkstra algorithm, minimizing congestion and providing continuous data transfer without interfering with other traffic flows. The proposed models have the capability to solve heterogeneous traffic by providing a combination of advanced traffic classifying and intelligent routing algorithms in SDNs, as shown in Figure 2. These interrelated models reduce network congestion as well as enhance system throughput and performance. Inter-related models are scalable and applicable for contemporary data centers and large-scale SDN architectures, and the floor for a more expressive and agile traffic management strategy.

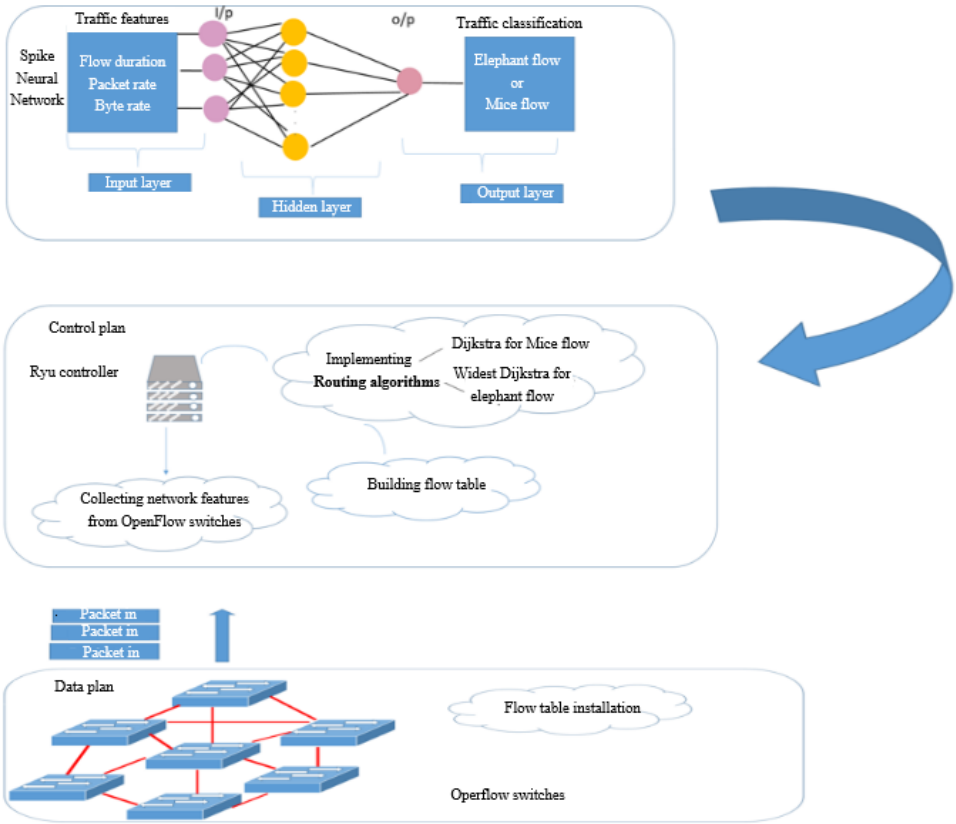


Figure 1. The proposed models

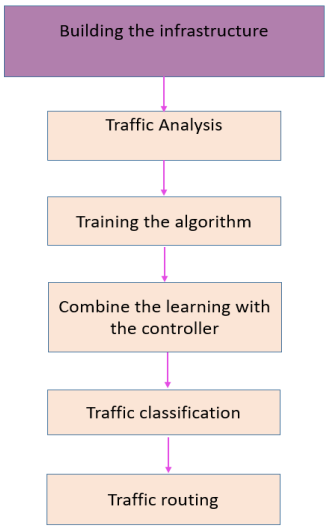


Figure 2. Overview of the workflow

3.1 Feature type selection stage

There are hundreds (potentially thousands) of applications that change all the time. Making it hard to categorize traffic at the application level. To identify traffic well, the proposed models have the power to identify elephants or mice on the network layer. Mice traffic is mostly query (e.g., Google searches and Facebook updates). A major portion of software-defined data center network (SDDCN) traffic is query traffic with a relatively smaller data transfer volume. Most of the time, the big upgrades come from the elephant traffic, like antivirus, updates, and movie downloads. According to Cisco, an elephant flow contains more than 15 packets, and a short flow contains fewer than 15 packets. The proposed models also use the byte size for referring to the flow, e.g., elephants or mice. Elephant flow usually gets 500 bytes or more for packets. In OpenSwitch, the mice traffic in data centers is limited to 10 KB. However, the actual traffic of SDDCN networks is diverse in nature. As a result, the characteristics used to distinguish between traffic classes are (flow duration, byte rate, and packet rate). The proposed models used the feature constructed for the Network Information Management and Security Group (NIMS) dataset, which has around 303,549 traffic flow statistics. It has two classes (Elephant class and Mice class), with 70% used for training and 30% for testing. Table 1 shows part of the NIMS dataset.

Table 1. Part of the NIMS dataset

Flow Size	Packet Size	Byte Counts	Traffic Class
120697	3	64	0
4096366	5	44	0
9289799	8	453	0
243187	4	490	0
4300478	63	65749	1
89872447	1188	76304	1
33494941	46290	1372596	1

Traffic class demonstrates the type of traffic where zero represents mice traffic and one represents elephant traffic.

3.2 Classification stage

The classification process concentrates on the SNNs, which act as a classifier. Figure 3 (Part A) depicts the fully linked feed-forward scheme of the SNN. The form of this structure consists of three layers: the input layer, the hidden layer, and the output layer. The output layer has one neuron that uses a spike mechanism to express traffic types such as elephants or mice. The hidden layer has twenty neurons, whereas the input layer contains three neurons that correlate to the traffic characteristic employed. In the SNN, each neuron has five connections (synapses) with variable delays and weights, as shown in Figure 3 (Part B). Trial and error determine the count of hidden neurons and synapses. Each neuron follows a three-step computational process. First, the membrane potential is formed by adding all input spikes together. The potential is then evaluated to see if it exceeds a certain threshold with value 1, which was determined by trial and error, as illustrated in Figure 3 (Part C). If the threshold is surpassed, the neuron will send a spike at a time, which is called the firing time (ft), and reset the membrane potential to zero, as shown in Figure 3 (Part C). For ease of categorization, the system has three major components: encoding and decoding functions, neuron model functions, and a modified learning algorithm.

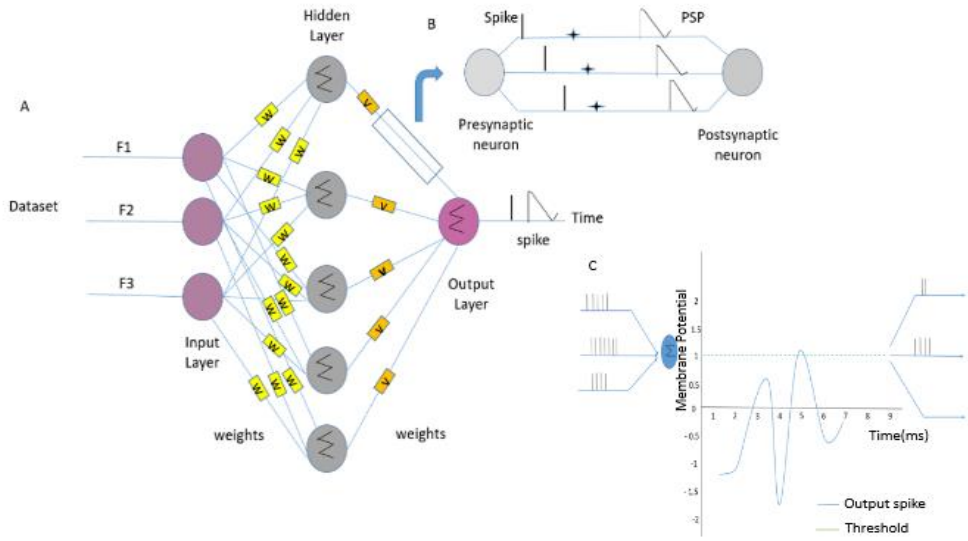


Figure 3. SNN architecture (A) SNN feedforward; (B) The connection between two neurons; (C) The computational phases of spiking neuron

3.3 Encoding and decoding operation

SNN operates on pulse information instead of actual data, as shown in Figure 4. The first step in developing an SNN is to transform analog input data into spike trains. Therefore, Eq. (1) is applied to encode the input values as spike timings.

$$t_h^f = T_{max} - round\left(T_{min} + \frac{(R_{in}-R_{min})(T_{max}-T_{min})}{R_{max}-R_{min}}\right) \quad (1)$$

T_{max} and T_{min} are the maximum and minimum intervals, respectively. R_{max} is a value more than the highest value in the input, while R_{min} is a value lower than the lowest value in the input. R_{in} reflects the current real data. *round* is an operation that produces a rounded value.

After training, apply the decoding Eq. (2) to transform the network's output spike time information to actual data [25].

$$RI\left(t_y^f\right) = \left(\frac{t_{y-R_{min}(R_{max}-R_{min})}^f}{T_{max}-T_{min}}\right) + R_{min} \quad (2)$$

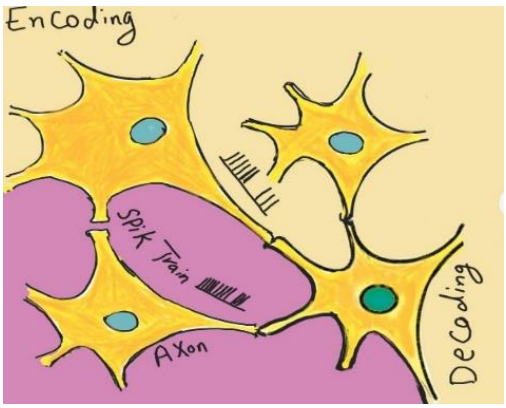


Figure 4. Methodology of encoding and decoding neuron model function

Several types of spiking neuron models have been proposed. The following models are more physiologically plausible: the Spike Response Model (SRM), the Izhikevich Model, the Hodgkin-Huxley Model (HH), and the Integrate and Fire Model. These models operate with SNNs. Choosing a suitable model relies on the user's needs [26, 27]. In this paper, the SRM is utilized due to its simple mathematics [28]. This model represents the SRM's link between input spikes and membrane potential. In this work, the function of the hyperbolic tangent is utilized in the SRM. When each spike arrives, a postsynaptic potential (PSP) is stimulated in the neuron if its membrane potential is under a particular threshold (θ) [29]. Membrane potential is the sum of all PSPs stimulated by all incoming spikes. The weights of synapses that convey these spikes also impact the value of PSP. The equation shows how to compute the PSP using the spike response function $\varepsilon(t)$ [30].

$$\varepsilon(t) = \begin{cases} 0, & t \le 0 \\ \tanh\left(\frac{t}{\tau_s}\right), & t > 0 \end{cases} \quad (3)$$

where,

$\varepsilon(t)$: Spike response function (SRF)

τ_s: Time decay constant with value 2

The derivative of $\varepsilon(t)$ is described below:

$$\frac{\partial \varepsilon}{\partial t} = \frac{1}{\tau_s}\left(1 - tanh^2(t/\tau_s)\right) \quad (4)$$

3.4 Learning method

Once real data is encoded into spike timings, the forward step begins. The second stage is the feedforward operation, which determines whether every neuron in the hidden layer is stimulated or not. Neurons can only spike once when their membrane potential (mp) surpasses the threshold value (ϑ). If the neuron is spiked, the algorithm moves to the next neuron in the same layer; the SNN algorithm calculates the membrane potential $mp_h(t)$ using Eq. (5) [31].

$$mp_h(t) = \sum_{x=1}^{X}\sum_{k=1}^{K} w_{xh}^k(t)\varepsilon(t - t_x^f - d^k) \quad (5)$$

X denotes the number of input neurons, and K is the number of synapses that link two neurons. w_{xh}^k indicates the synapse weight coefficient for presynaptic and postsynaptic neurons. The presynaptic neurons' spike time is t_x^f, while the Synaptic delay is represented by d^k.

When the neurons of the hidden layer are complete, the algorithm moves to the output layer and performs the same manner, except in this case, the spikes of the hidden layer are inputs to the output neurons. Then the total error is computed using Mean Square Error (MSE), as shown in Eq. (6) [32].

$$MSE = \frac{1}{2}\sum_{y\epsilon Y}\left(t_y^f - t_y^d\right)^2 \quad (6)$$

where, the t_y^f represents the spiking firing time, while t_y^d represents the desired spiking.

The reverse phase of the process involves adjusting the weights of synapses for connections. SNN is trained by updating the weight of each synapse using the backpropagation method (SPIKEPROP) to decrease the MSE value. Backpropagation begins in the output layer and returns to the hidden layer.

Synapses weights among the outcome and hidden neurons are modified using Eqs. (7)-(9).

$$\delta_y = \frac{t_y^d - t_y^f}{\sum_{h=1}^{H}\sum_{k=1}^{K} w_{hy}^k \frac{\partial}{\partial t} y_h^k} \quad (7)$$

$$\Delta w_{hy}^k = \alpha\, \delta_y y_h^k \quad (8)$$

$$w_{hy}^k(t+1) = w_{hy}^k(t) - \Delta w_{hy}^k \quad (9)$$

where, δ_y delta function applies to the outcome layer.

Synapse weights among hidden neurons and the input neuron is adjusted using the Eqs. (10)-(12).

$$\delta_h = \frac{\sum_{h=1}^{H}\delta_y \sum_{k=1}^{K} w_{hy}^k(t)\frac{\partial}{\partial t} y_h^k}{\sum_{x=1}^{X}\sum_{k=1}^{K} w_{xh}^k(t)\frac{\partial}{\partial t} y_x^k} \quad (10)$$

$$\Delta w_{xh}^k = \alpha.\, \delta_h y_x^k \quad (11)$$

$$w_{xh}^k(t+1) = w_{xh}^k(t) - \Delta w_{xh}^k \quad (12)$$

where, δ_h is the delta function of hidden neurons and α represents the learning rate with value 0.01.

3.5 Traffic routing stage

Routing is a critical element in networking that involves configuring devices and creating network policies to facilitate the transfer of data from point A to point B. At its most basic level, routing manages the flow of data packets to their destination, taking into account network devices such as switches and routers to ensure they are delivered optimally. This process is the routing algorithm which plays an important role in identifying the optimal path for data packets to travel all over the network. These algorithms take into consideration several variables, including the topology of the network, link capacity, traffic load, latency, and dependability in order to discover the optimal path. In doing so, they ensure data is delivered effectively through the use of lower latency and higher throughput. Modern routing algorithms are not only efficient but also intended to improve transmission reliability by circumventing congested or faulty paths. They also facilitate a network to become more scalable by not only tolerating the growth of complex and rapid networks like data centers and wide-area of networks but also adapting to the modification of network structure and largeness, including, but not limited to, advanced routing techniques that cater to varying traffic demands, including load aware algorithms (e.g., Widest Path) and shortest path algorithm (e.g., Dijkstra). Besides maximizing resource utilization, these approaches also address the needs of different types of traffic – bandwidth-intensive and latency-sensitive flows. Even as networks advance, reliable and effective communication still relies on routing as a fundamental component across ever more elaborate networking circumstances. After finishing both the learning phase and traffic classification, based on the classification results, the proper routing algorithms are executed. For the MF traffic, these are short-lived and latency-sensitive; thus, the Dijkstra algorithm will be utilized. The algorithm helps in finding the shortest path from the source to the destination and it provides low latencies and quickly delivers packets of data. In contrast, for Elephant flow traffic with large bandwidth requirements and long lifetimes, the Widest Dijkstra algorithm is used. These variant places more weight on paths with larger bandwidth capacity, which lessens the probability of congestion and accommodates the processing of heavy traffic loads. The Dijkstra algorithm is a core component of SDN efficiency and performance. The shortest path computation in a network increases the routing efficiency and thereby the overall QoS. The platform facilitates core network elements like low latency and high reliability by reducing delays and optimizing path selection. Finally, the algorithm helps distribute the server load by balancing the traffic on the network, preventing bottlenecks in the connection, and ensuring a smoother data flow. Further refinement of this approach is achieved by the incorporation of the Widest Dijkstra algorithm for EF—taking into account the specific difficulties that bandwidth-intensive flows introduce. The system enhances the responsiveness of MFs while optimizing the resource usage of EFs by dynamically allocating resources according to flow characteristics. These routing strategies work together to ensure that the SDN runs at optimal performance, managing heterogeneous traffic and increasing the scalability and reliability of modern network environments. The SDN routing approach is decomposed into two general components: obtaining a global network view and realizing this view in practice.

3.6 Obtaining a global perspective of the network

The first phase consists of collecting network data to determine the topology, building a base for intelligent and efficient routing decisions. SDN separates the control and administration planes and the data-forwarding components, unlike traditional routing protocols such as Open Shortest Path First (OSPF). In this method, a centralized SDN controller makes routing decisions, eliminating the built-in knowledge of the network within forwarding devices (e.g., switches). For this, the controller requires real-time and accurate information about the networks, such as the statuses of dynamic links and the topological structure. Topology discovery works through Link Layer Discovery Protocol (LLDP) using the SDN controllers. LLDP is good for collecting static link status data such as switch connection, since this data does not change a lot over time. Through continuous quote LLDP exchanges, the controller builds an intake view of the entire physical network topology, including all nodes and links. But in dynamic scenarios, routing needs more than static measurements. The capabilities of dynamic link-state metrics, such as capacity, latency, and available bandwidth are susceptible to real-time traffic situations. An SDN controller must constantly monitor these dynamic characteristics in order to ensure correct Network State Information (NSI). This involves retrieving updated link-state metrics from frequent queries of the underlying switches, for each connection. If the SDN controller is cognizant of not only static topology but also dynamic link states, then exploit this information to make judicious routing choices. In order to utilize resources more efficiently, reduce congestion, and meet QoS requirements, the controller dynamically changes paths by combining real-time link-state information with the static topology map provided by LLDP. SDN proves its flexibility and dominance in the ability to manage modern, complex networking environments by maintaining a central, real-time view of the network.

3.7 Routing algorithms in SDN

In SDN, routing algorithms are responsible for making the right decision about which path - or paths - traffic will flow, for example. After this path is established, the controller then modifies the forwarding devices to make sure that the packets are sent the right way. Unlike conventional networks, where routing decisions are spread throughout switches in an SDN network, they only need to be made at the controller. SDN-based shortest path finding involves finding a shortest path from source to destination using Dijkstra's algorithm, choosing the optimal path based on various attributes of the network like Latency, Bandwidth, and Hop count. An important component of this is the SDN controller which absorbs all the analytics from each and every switch, bandwidth readily available, which connections are up, and more performance metrics. The Dijkstra algorithm generates a weighted graph to model the network. This graph depicts switches along with other network devices shown as nodes. Edges represent the relationships or connections existing between these nodes. Each edge is given a weight according to the cost of traversing the link, determined through metrics such as latency and bandwidth. Then, Dijkstra’s Algorithm is used to find the shortest or least cost path between the source and the destination nodes on the graph. The method finds the minimum cumulative cost because it iteratively selects the next node with the lowest

weight until it arrives at the destination. Here, the cost of traversing the connection is mathematically expressed as in Eq. (13):

$$Cij = Lij + Bij + Hij \quad (13)$$

where,

Cij: The link's cost between nodes

Lij: The link's latency

Bij: The link's available bandwidth

Hij: Hop count for the link

After the best path is calculated, the SDN controller processes the data into forwarding rules and sends them to the correlated switches. As each packet moves along its chosen path, it has to follow certain rules that guide the switches to forward it correctly, providing optimal data transfer and quality of service compliance. Reinforcing this dynamic approach, SDN could keep high performance in several dynamic traffic conditions, optimize using the resources, and adapt to the real-time networking environment. The cost of a link is computed based on multiple matrices, including (available bandwidth, capability, latency, hop count, etc.). Then select the path that has the smallest latency and capability and consider it as the best shortest path. For the widest Dijkstra follows the same procedure and instead of choosing one shortest path, it selects multiple shortest paths to forward the elephant flow that needs many routes due to its large volume leading to enhancing its throughput as shown in the algorithm below. To illustrate that assume that a host (Host) wishes to transmit several packets to a server (Server) and that there are several routes between the Host and the Server, as illustrated in Figure 5. The following procedures will be used to compute the link cost of routes.

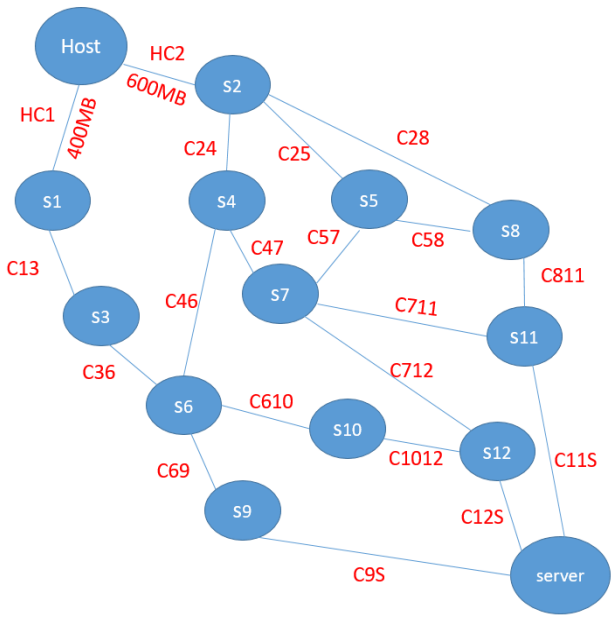


Figure 5. The map between the host and server

Find all Possible Routes:

First route R1 passes through switches S1-S3-S6-S9-Server

Second route R2 passes through switches S1-S3-S6-S10-S12-Server

Third route R3 passes through switches S2-S4-S6-S9-Server

Fourth route R4 passes through switches S2-S4-S6-S10-S12-Server

Fifth route R5 passes through switches S2-S4-S7-S12-Server

Sixth route R6 passes through switches S2-S4-S7-S11-Server

Seventh route R7 passes through switches S2-S8-S11-Server

Eighth route R8 passes through switches S2-S5-S8-S11-Server

Ninth route R9 passes through switches S2-S5-S7-S12-Server

Tenth route R10 passes through switches S2-S5-S7-S11-Server

Then calculate the available bandwidth, latency and capability for each route as:

Available Bandwidth for RI

AB1 = AB (HC1) + AB (C13) + AB (C36) + AB (C69) + AB (C9S)

Available Bandwidth for R2

AB2 = AB (HC1) + AB (C13) + AB (C36) + AB (C610) +AB (C1012) +AB (C12S)

Available Bandwidth for R3

AB3 = AB (HC2) + AB (C24) + AB (C46) + AB (C69) + AB (C9S)

Available Bandwidth for R4

AB4 = AB (HC2) + AB (C24) + AB (C46) + AB (C610) + AB (C1012) + AB (C12S)

Available Bandwidth for R5

AB5 = AB (HC2) + AB (C24) + AB (C47) + AB (C712) + AB (C12S)

Available Bandwidth for R6

AB6 = AB (HC2) + AB (C24) + AB (C47) + AB (C711) + AB (C11S)

Available Bandwidth for R7

AB7 = AB (HC2) + AB (C28) + AB (C811) + AB (C11S)

Available Bandwidth for R8

AB8 = AB (HC2) + AB (C25) + AB (C58) + AB (C811) + AB (C11S)

Available Bandwidth for R9

AB9 = AB (HC2) + AB (C25) + AB (C57) + AB (C721) + AB (C12S)

Available Bandwidth for R10

AB10 = AB (HC2) + AB (C25) + AB (C57) + AB (C711) +AB (C11S)

After that compute the latency for each route as follows:

Latency of R1 = 1/ AB1

Latency of R2 = 1/ AB2

Latency of R3 = 1/ AB3

Latency of R4 = 1/ AB4

Latency of R5 = 1/ AB5

Latency of R6 = 1/ AB6

Latency of R7 = 1/ AB7

Latency of R8 = 1/ AB8

Latency of R9 = 1/ AB9

Latency of R10 = 1/ AB10

Finally, compute the capability for each switch using the Eq. (14).

$$w[v] = \frac{\sum_{f \in Flow(v)} \text{Bits}(f)}{\text{capacity}(v)} \quad (14)$$

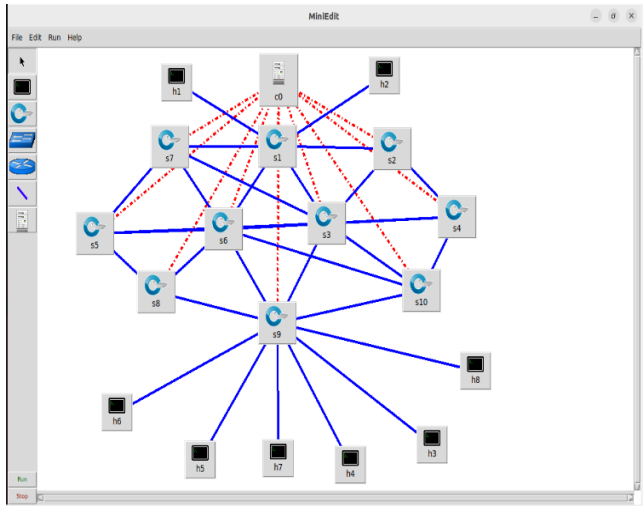
Then based on the results select the path that has the smallest latency and capability and consider it as the best shortest path.

Algorithm 1: Traffic Classification and Traffic Routing Suggested Models in Generic Form

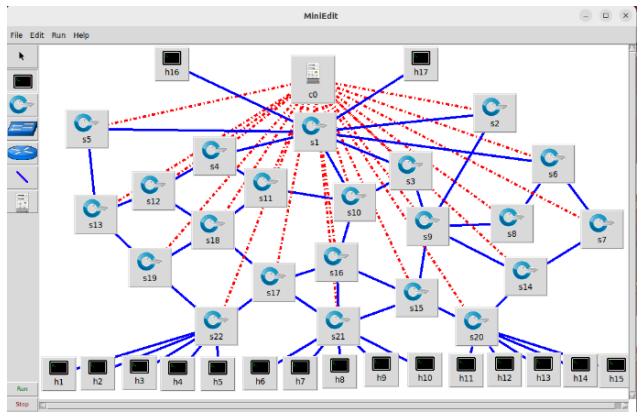
1: Begin
2: G(*): Classification using SNN
3: T(*): Traffic analysis
4: F(*): extraction of features from traffic analysis
5: E(*): Encoding the actual value of the extracted feature into the spike
6: EVO: represents Event Occurs
7: 0 represents the Mice class
8: 1: represents the elephant class
9: Y: represents Yes
10: N: represents No
11: FRE: represents Flow Rule Exists
12: Initialize The Traffic (End Host Makes Their Requests)
13: if EVO = N then
14: wait until EVO =Y
15: Else
16: SDN OFSs Receive the Packets and Check the Flow Table
17: if FRE = Y then
18: Forwarding Packets Based on Flow Rules
19: end
20: Else
21: Alarm the Controller that there is a flow need to be forward by sending a packet in msg
22: Continue flow processing in The Controller
23: Z ← T(flow)
24: * Z represents the analyzed traffic*\
25: Activate the SNN Classifier
26: M← F(Z)
27: * M is the extracted feature in a real number *\
28: S ← E(M)
29: * S is the extracted feature in spike time *\
30: W ← G(s)
19: * W is the outcome of the classes of the SNN classifier, with probable values of 0 or 1 *\
31:O←W
32: if O=0
33: * Consider the flow as Mice flow*\
34: Implement the Dijkstra Algorithm
35: Find the Shortest Path from Source to Destination based on Specific QoS
36: else
37: *The flow is classified as Elephant flow*\
38: Implement the Wideset Dijkstra Algorithm
39: Find the Multiple Shortest Paths from Source to Destination based on Specific QoS
40: The Controller Makes the Flow Rules
41: Install the Flow Rule on OFSs
42: Forwarding Packets Based on Flow Rules
43: End

4. PERFORMANCE METRICS OF THE PROPOSED MODEL

In this paper, the Mininet emulation framework is employed to develop and build a realistic SDN environment on a virtual machine [33-36]. Due to its construction of lightweight virtualization, Mininet offers great flexibility and facilitates the development of a dynamic and scalable network topology tailored to the proposed models. In this paper, two scenarios were designed to test the performance of the proposed traffic models on different traffic conditions. The first scenario topology, as shown in Figure 6(a), has two servers that work as traffic sources and sinks, one SDN controller, ten OpenFlow switches composing the network core and edge layers, and six hosts that are spread through the topology. The small scenario topology is also implemented as shown in Figure 6(b). These setups replicated an actual DCN environment and thus simulated realistic traffic flows and network functionalities. The flexible Mininet environment facilitated the generation and control of varied traffic types, enabling extensive investigations into the performance of the overall SNN classification and the routing conducted via both the Dijkstra and Widest Dijkstra algorithms within the same emulation testbed. These regulated yet realistic environments provided by this emulation setup allowed for the validation of the suggested approaches' scalability, effectiveness, and effect on network performance.



(a)



(b)

Figure 6. (a) Custom first topology of the network; (b) Custom second topology of the network

The Ryu controller is used due to its programmability and wide support for SDN portability. Evaluating the success of a classification model and its learning process requires a comprehensive examination of the important metrics that represent its efficiency. Evaluating the success of a routing model necessitates a thorough examination of important parameters that represent its efficiency, reliability, and adaptability. The suggested routing model is evaluated based on the following essential performance characteristics:

Throughput, bandwidth utilization, packet loss, and latency. The evaluation compares the performance of the SDN-Ryu controller and the approach described in the cited study [22], which employs K-means clustering for traffic categorization, to the proposed system based on the Data Center Network architecture. As shown in Figure 7, the comparison focuses on two sorts of fluxes: EFs (big, long-lived flows) and MFs (small, short-lived flows). Throughput, an important statistic in SDN traffic routing, is used as the major parameter in this assessment since it directly shows the number of packets successfully received throughout the simulation time. The evaluation findings show that the suggested model outperforms both the Ryu controller and the paper approach in terms of throughput for two flow types by 60% and 20% respectively. These benefits demonstrate the model's capacity to manage a variety of traffic types more successfully than standard SDN control approaches. As the number of parallel flows grows, throughput often falls owing to increased network congestion. However, the suggested technology outperforms both the Ryu controller and the paper method in terms of throughput on average. This shows that the suggested model not only adapts better to larger traffic loads but also improves network efficiency under challenging conditions.

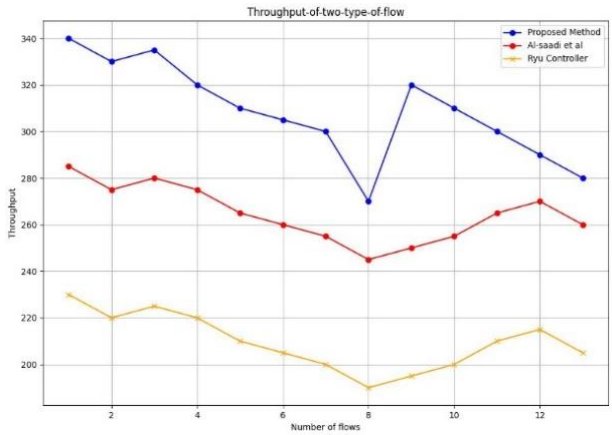


Figure 7. The comparison of the proposed method to the Ryu controller and cited paper in terms of throughput for two types of flows

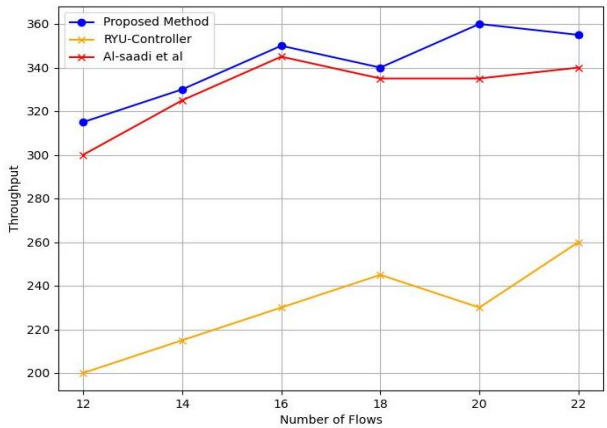


Figure 8. The comparison of the proposed method to the Ryu controller and the cited paper for elephant flow

Figure 8 compares the throughput performance for EFs using the recommended strategy to the Ryu controller and the method given in the cited paper. The findings clearly show that, on average, the suggested strategy outperforms both the Ryu controller and the paper's solution. This higher performance is due to the optimized flow classification and routing strategy contained in the proposed technology. Throughput amounts change owing to the dynamic nature of the routing operation, which is impacted by the unique properties of each flow. Specifically, the routing algorithm in the proposed approach chooses the best path for each representative flow, taking into consideration characteristics such as flow size and duration. On average, the proposed model provides an estimated 60% improvement in throughput over the Ryu controller and a 10% gain over the paper technique for EFs. This significant gain demonstrates the proposed technique's capacity to more efficiently prioritize big, long-lived flows (EFs), hence increasing data transmission speeds and overall network efficiency.

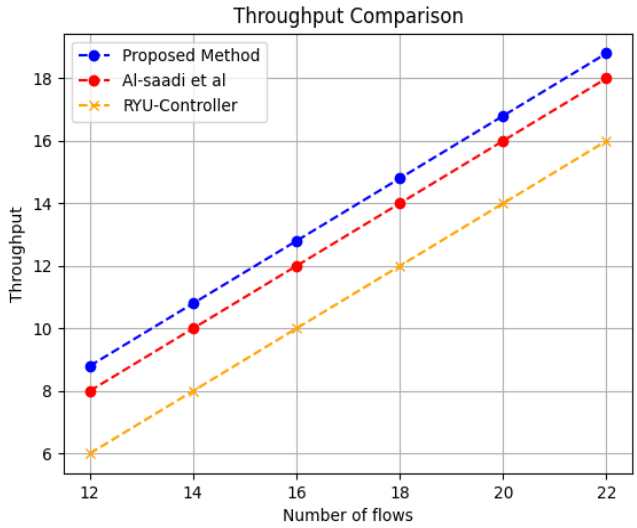


Figure 9. The comparison of the proposed method to the Ryu controller and cited paper in terms of mice flow throughput

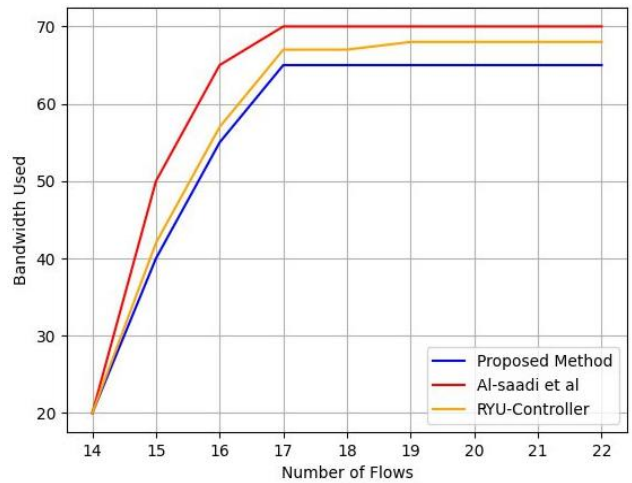


Figure 10. The comparison of the bandwidth utilized by the proposed model to the Ryu controller and cited paper for both types

The throughput performance for MFs (small, transient flows) utilizing the suggested model in comparison to the Ryu controller and the technique described in the cited paper is shown in Figure 9. The findings show that the suggested model outperforms the Ryu controller and the SDN-based solution in the cited study [22], consistently delivering greater throughput

across all MFs. In particular, the proposed model outperforms the paper's approach and the conventional SDN configuration in terms of throughput by about 50% and 12% respectively. The main reason for this improvement is the model's improved flow classification and routing system, which is designed to manage the particularities of mice fluxes. MFs, as opposed to EFs, require quick processing and effective routing in order to avoid bottlenecks and guarantee the timely delivery of tiny data packets. Alongside throughput, another important metric for assessing network efficiency is bandwidth consumption. Figure 10 shows that the proposed model not only performs well in throughput but also in terms of bandwidth usage when compared to the Ryu controller and the method described in the cited paper. In particular, the proposed model consistently uses less bandwidth for the majority of flows while maintaining high throughput levels, which is due to the model's sophisticated traffic classification. The proposed model's categorization techniques allow it to manage these flows more effectively than conventional SDN techniques, which frequently over-allocate resources, resulting in increased bandwidth usage. Furthermore, it is remarkable that throughput and network performance are not sacrificed in order to achieve this increase, even if the decreased bandwidth utilization may appear expected given the model's emphasis on classification and adaptive routing. Rather, the model optimizes in a balanced way, optimizing throughput while consuming the fewest resources possible. This effective use of bandwidth is especially helpful in situations where network resources are expensive or scarce, including in cloud environments, data center networks, and Internet of Things deployments. The suggested model reduces bandwidth consumption over the paper's approach and the conventional SDN configuration by about 7% and 5% respectively, which frees up resources for more flows while also lowering operating costs.

Moving to the second topology, the proposed model conducted two exhaustive tests in relation to the manner discussed in previous study [22] in order to evaluate the mechanism above against the Ryu controller and the proposed mechanism. The assessment was done on the same parameters and conditions as the first topology to maintain consistency and fairness when comparing. Figures 11 and 12 show throughput and bandwidth utilization metrics acquired during tests for the proposed technique, the Ryu controller, and the paper's method. These figures depict how the strategies were performed under various scenarios.

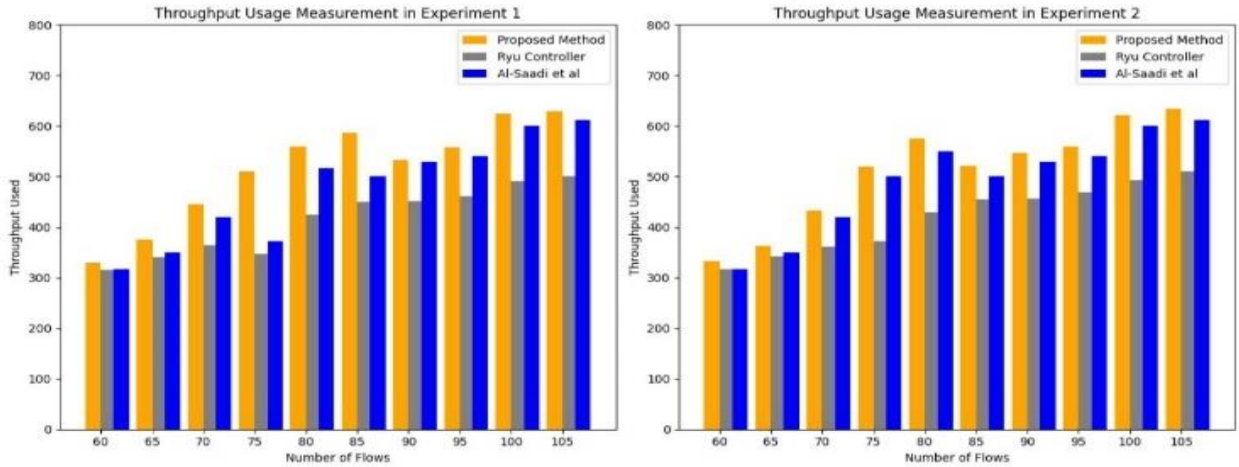


Figure 11. A comparison between the suggested approach to the Ryu controller and the cited paper for both types of flows in the two experiments in terms of throughput

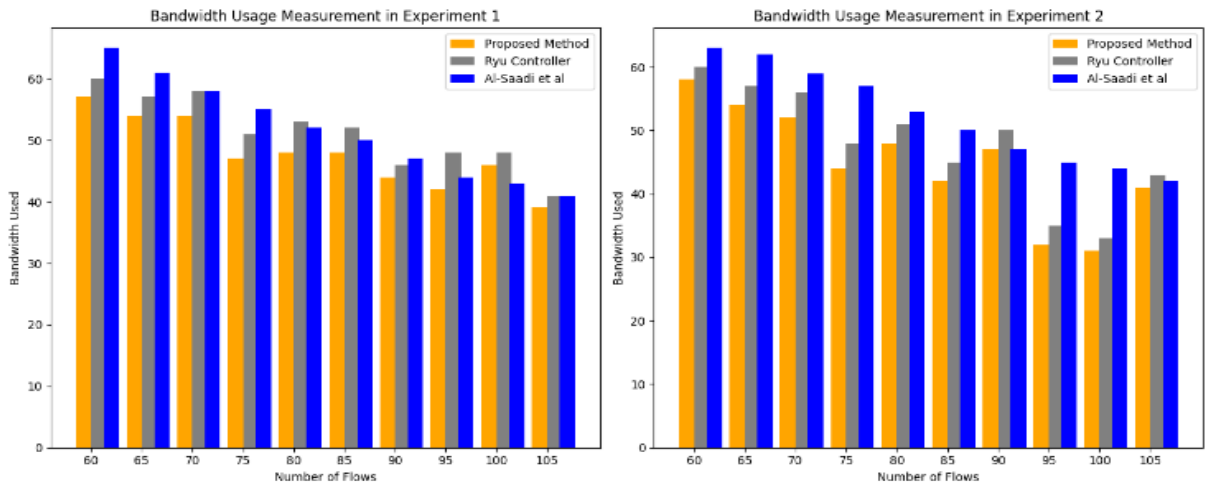


Figure 12. A comparison between the suggested approach to the Ryu controller and cited paper for both types of flows in the two experiments in terms of bandwidth

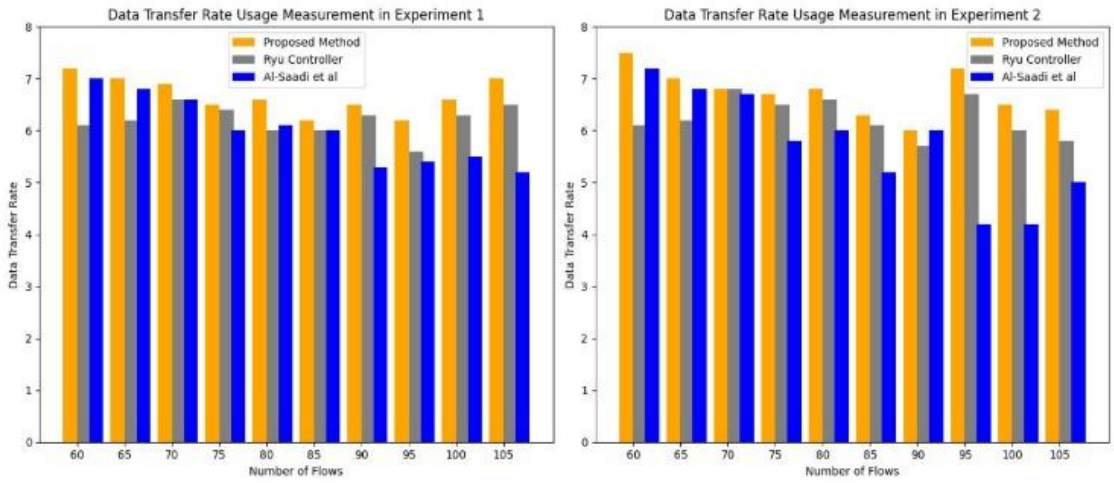


Figure 13. A comparison of two types of flows in two experiments using the proposed method, the cited paper, and the Ryu controller of data transfer rates

In Experiment 2, the suggested mechanism consistently outperformed both the Ryu controller and the aforementioned approach in throughput and bandwidth utilization. The charts in Figure 13 provide a clear illustration of this by showing the comparison of results across different flows. The proposed mechanism provided a higher throughput and a more efficient bandwidth utilization, which indirectly reflected the ability to handle the network traffic more appropriately. The performance enhancement was consistent across flows, which in turn demonstrates the strength of this proposed method and its adaptation to many situations. Thus, the proposed mechanism achieved better performance than Ryu controller while improving the entire network performance significantly. These improvements show the potential of the described technique to be a scalable and conditional yet reliable method for controlling the network resource in dynamic and overloaded scenarios.

The data transmission rate represented the third fundamental measure used to evaluate the performance of the proposed process with respect to both the Ryu controller and the technique explained in the reference paper. It was supplied explicitly to give us an idea about how well the proposed method regulates network data transfer. The proposed strategy is significantly more efficient than the Ryu controller and the method provided in the referenced paper in terms of data transfer for most of the flows, as illustrated in Figure 13. It proves that in terms of transmission, the proposed method can make the best use of the relevant resources of the network, and also the transmission delay and packet loss can be controlled within a limited range. The remaining shows the performance pattern across data transmission rate measurements from the second experiment, which is similar to those given for throughput and bandwidth utilization. The proposed method dominates the Ryu controller and the method in previous study [21] for most of the flows by 15% and 30% respectively, which proves the better performance of the proposed method under dynamic network topology and dynamic traffic requirement.

Packet loss plays a significant role in determining network performance under a request environment that requires both integrity and speed of data being transferred. Low PL indicates a more dependable network with less packet drop, as in the case of latency-sensitive applications, it is necessary to maintain quality of service. As shown in Figure 14, the packet loss is greatly reduced compared to a normal SDN controller system, with the proposed method performed by 50%. The improvement comes from a feedback control system that is complex and embedded in the proposed model, where SNNs are used to control traffic. The SNN-based controller smartly prioritizes every traffic stream, determines the optimal route, and safeguards against packet drops. In contrast to the classic SDN controller, which might still use static or less adaptive techniques, the SNN technique continues to learn and adapt to the changing traffic flows. Ensures that even in the presence of high network load, packets are transmitted with minimal latency, and packet drops are significantly minimized. Suitable only for the most critical, much-needed services requiring true reliability and packet delivery in real-time, the proposed model appears to be capable of dynamically adapting to changes in network topology while minimizing packet loss. Such a benefit is important for ensuring that service-level agreements (SLAs) are met, thus increasing service satisfaction and maintaining trust that the network will continue to function.

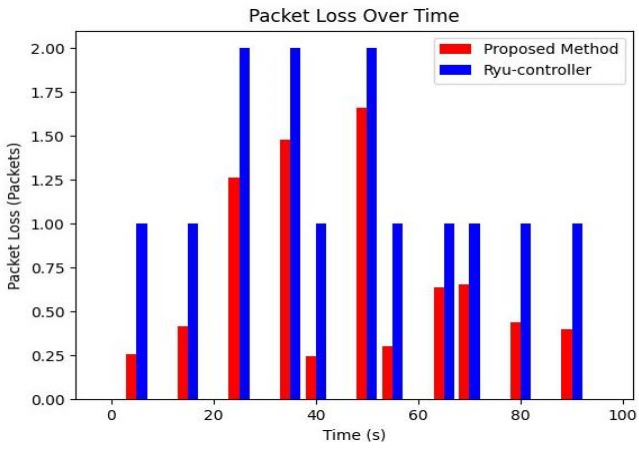


Figure 14. A comparison of the proposed method to the Ryu controller in terms of packet loss

As shown in Figure 15, the delay results prove that the proposed model for the traffic classification and routing is much better than the traditional Ryu controller regarding a reduction in the network delay. It is a key performance indicator of the time taken by data packets to travel from source to destination. The optimized flow classification and

adaptive routing mechanisms adopted in the proposed model give rise to a lower delay in the model than in the Ryu controller by 60%. The dynamic approach helps to ensure that all types of traffic are handled in the best possible way at every point, optimally avoiding congestion and thus supporting lower overall delay. On the other hand, Ryu is a more generic controller that does not have traffic type-aware routing capabilities, causing a significant waste of resource utilization of the network, as more and more long-lived, large size EFs obtain a good quality of service of network while starving the others and causing a bottleneck that increases congestion and delay of all types of traffic. Hence, the proposed approach is more optimal than the Ryu controller and indicates that it can identify the required path with respect to flow type, as shown in Table 2.

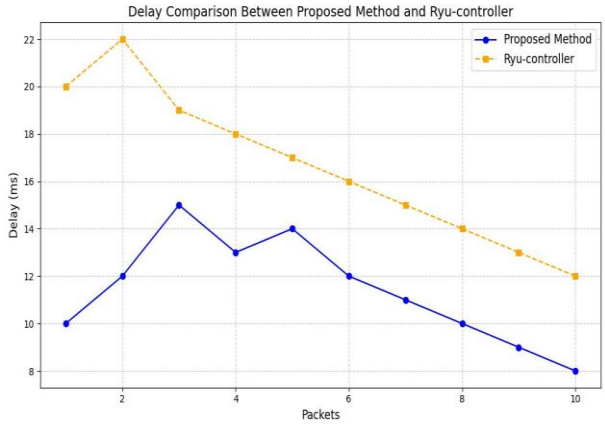


Figure 15. A comparison of the proposed method to the Ryu controller in terms of delay

Table 2. The overall performance of the proposed model compared to Ryu-Controller and Al-Saadi et al. [22]

Performance Metrics	The Proposed Method Enhancement vs. the Ryu-Controller	The Proposed Method Enhancement vs. Al-Saadi et al. [22]
Throughput for two types	60%	20%
Throughput for elephant flow	60%	10%
Throughput for mice flow	50%	12%
Bandwidth usage	5%	7%
Data transfer rate	15%	30%
Packet loss	50%	
Latency	60%	

5. CONCLUSIONS

The SDN enhancements in this paper are divided into two parts: the traffic classification part and the traffic routing part, which aim to enhance the efficiency and performance of this technology. The initial part of the paper presents an SDN-based classification model utilizing a single SNN under the supervised learning paradigm based on flow characterization and classification. The proposed method illustrates the potential of SNNs for analyzing and classifying network traffic into EF and MF categories with respect to their performance traits (e.g., duration, packet size, and byte rate). Part two of the paper includes how to route traffic by classification type. MFs have short durations and are latency-sensitive, so the shortest path is necessary for MFs, and the Dijkstra algorithm is considered because it ensures faster performance with less latency time. The Widest Dijkstra algorithm can be applied to EFs, which are often bandwidth-intensive and long-lived, to assign paths with the widest links first to reduce congestion and use the bandwidth more effectively. This double-routing approach extends the SDN's traffic diversity by allowing the selection of different routing actions based on each flow specification. These approaches significantly increase the global performance and efficiency of the SDN system, guaranteeing a proper usage of its resources while still respecting the QoS requirements through heterogeneous traffic. The models used have proven their efficiency by outperforming the traditional Software Defined Network and the K-means algorithm in terms of increasing throughput by 60% and 20%, enhancing bandwidth utilization by 5% and 7%, reducing packet loss by 50%, and decreasing latency by 60%, respectively. Finally, this research presents a promising work for using SNN-based traffic classification and adaptive traffic routing, where SDN can be useful. To gain even more from this work, it can be integrated with the concept of many controllers to optimize the benefits of both concepts and provide enormous scalability for SDN, which we are working on in the near future.

REFERENCES

[1] Ali, T.E., Morad, A.H., Abdala, M.A. (2020). Traffic management inside software-defined data centre networking. Bulletin of Electrical Engineering and Informatics, 9(5): 2045-2054. https://doi.org/10.11591/eei.v9i5.1928

[2] Darabseh, A., Al-Ayyoub, M., Jararweh, Y., Benkhelifa, E., Vouk, M., Rindos, A. (2015). SDDC: A software defined datacenter experimental framework. In 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, Italy, pp. 189-194. https://doi.org/10.1109/FiCloud.2015.127

[3] Thirupathi, V., Sandeep, C.H., Kumar, N., Kumar, P.P. (2019). A comprehensive review on SDN architecture, applications and major benifits of SDN. International Journal of Advanced Science and Technology, 28(20): 607-614.

[4] Al-Kaseem, B.R., Al-Raweshidy, H.S. (2017). SD-NFV as an energy efficient approach for M2M networks using cloud-based 6LoWPAN testbed. IEEE Internet of Things Journal, 4(5): 1787-1797. https://doi.org/10.1109/jiot.2017.2704921

[5] Islam, M.S., Al-Mukhtar, M., Khan, M.R.K., Hossain, M. (2023). A survey on SDN and SDCN traffic measurement: Existing approaches and research challenges. Eng, 4(2): 1071-1115. https://doi.org/10.3390/eng4020063

[6] Priyadarsini, M., Bera, P. (2021). Software defined networking architecture, traffic management, security, and placement: A survey. Computer Networks, 192: 108047. https://doi.org/10.1016/j.comnet.2021.108047

[7] Mendiola, A., Astorga, J., Jacob, E., Higuero, M. (2016). A survey on the contributions of software-defined networking to traffic engineering. IEEE

Communications Surveys and Tutorials, 19(2): 918-953. https://doi.org/10.1109/COMST.2016.2633579
[8] Wang, N., Ho, K.H., Pavlou, G., Howarth, M. (2008). An overview of routing optimization for internet traffic engineering. IEEE Communications Surveys and Tutorials, 10(1): 36-56. http://doi.org/10.1109/COMST.2008.4483669
[9] Yusuf, M.N., bin Abu Bakar, K., Isyaku, B., Saheed, A.L. (2023). Review of path selection algorithms with link quality and critical switch aware for heterogeneous traffic in SDN. International Journal of Electrical and Computer Engineering Systems, 14(3): 345-370. https://doi.org/10.32985/ijeces.14.3.12
[10] Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R. (2009). The nature of data center traffic: Measurements and analysis. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, Illinois, United States, pp. 202-208. https://doi.org/10.1145/1644893.1644918
[11] Isyaku, B., Mohd Zahid, M.S., Bte Kamat, M., Abu Bakar, K., Ghaleb, F.A. (2020). Software defined networking flow table management of openflow switches performance and security challenges: A survey. Future Internet, 12(9): 147. https://doi.org/10.3390/fi12090147
[12] Li, G., Wang, X., Zhang, Z. (2019). SDN-based load balancing scheme for multi-controller deployment. IEEE Access, 7: 39612-39622. https://doi.org/10.1109/ACCESS.2019.2906683
[13] Zaher, M., Alawadi, A.H., Molnár, S. (2021). Sieve: A flow scheduling framework in SDN based data center networks. Computer Communications, 171: 99-111. https://doi.org/10.1016/j.comcom.2021.02.013
[14] Shirali-Shahreza, S., Ganjali, Y. (2018). Delayed installation and expedited eviction: An alternative approach to reduce flow table occupancy in SDN switches. IEEE/ACM Transactions on Networking, 26(4): 1547-1561. https://doi.org/10.1109/TNET.2018.2841397
[15] Liu, W.X., Cai, J., Chen, Q.C., Wang, Y. (2021). DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks. Journal of Network and Computer Applications, 177: 102865. https://doi.org/10.1016/j.jnca.2020.102865
[16] Kumar, S., Bansal, G., Shekhawat, V.S. (2020). A machine learning approach for traffic flow provisioning in software defined networks. In 2020 International Conference on Information Networking, Barcelona, Spain, pp. 602-607. https://doi.org/10.1109/ICOIN48656.2020.9016529
[17] Modi, T.M., Swain, P. (2023). Enhanced routing using recurrent neural networks in software defined-data center network. Concurrency and Computation: Practice and Experience, 35(5): e7557. https://doi.org/10.1002/cpe.7557
[18] Mu, T.Y., Al-Fuqaha, A., Shuaib, K., Sallabi, F.M., Qadir, J. (2018). SDN flow entry management using reinforcement learning. ACM Transactions on Autonomous and Adaptive Systems, 13(2): 1-23. https://doi.org/10.1145/3281032
[19] Lin, C.Y., Chen, C., Chang, J.W., Chu, Y.H. (2014). Elephant flow detection in datacenters using openflow-based hierarchical statistics pulling. In 2014 IEEE Global Communications Conference, Texas, United States, pp. 2264-2269. https://doi.org/10.1109/GLOCOM.2014.7037145
[20] Aymaz, Ş., Çavdar, T. (2023). Efficient routing by detecting elephant flows with deep learning method in SDN. Advances in Electrical and Computer Engineering, 23(3): 57-66. https://doi.org/10.4316/AECE.2023.03007
[21] Yusuf, M.N., Bakar, K.B.A., Isyaku, B., Osman, A.H., Nasser, M., Elhaj, F.A. (2023). Adaptive path selection algorithm with flow classification for software-defined networks. Mathematics, 11(6): 1404. https://doi.org/10.3390/math11061404
[22] Al-Saadi, M., Khan, A., Kelefouras, V., Walker, D.J., Al-Saadi, B. (2023). SDN-based routing framework for elephant and mice flows using unsupervised machine learning. Network, 3(1): 218-238. https://doi.org/10.3390/network3010011
[23] Shears, O., Yazdani, A. (2020). Spiking neural networks for image classification. Virginia Polytechnic Institute and State University, 2(6): 1-7. https://doi.org/10.13140/RG.2.2.27001.80486
[24] Abubaker, B.A., Ahmed, S.R., Guron, A.T., Fadhil, M., Algburi, S., Abdulrahman, B.F. (2023). Spiking neural network for enhanced mobile robots' navigation control. In 2023 7th International Symposium on Innovative Approaches in Smart Technologies, Istanbul, Turkey, pp. 1-8. https://doi.org/10.1109/ISAS60782.2023.10391395
[25] Al-Jamali, N.A.S., Al-Raweshidy, H.S. (2020). Modified Elman spike neural network for identification and control of dynamic system. IEEE Access, 8: 61246-61254. https://doi.org/10.1109/ACCESS.2020.2984311
[26] Yellakuor, B.E., Moses, A.A., Zhen, Q., Olaosebikan, O.E., Qin, Z. (2020). A multi-spiking neural network learning model for data classification. IEEE Access, 8: 72360-72371. https://doi.org/10.1109/ACCESS.2020.2985257
[27] Yamazaki, K., Vo-Ho, V.K., Bulsara, D., Le, N. (2022). Spiking neural networks and their applications: A review. Brain Sciences, 12(7): 863. https://doi.org/10.3390/brainsci12070863
[28] Oniz, Y., Kaynak, O., Abiyev, R. (2013). Spiking neural networks for the control of a servo system. In 2013 IEEE International Conference on Mechatronics, Kagawa, Japan, pp. 94-98. https://doi.org/10.1109/ICMECH.2013.6518517
[29] Hu, R., Chang, S., Wang, H., He, J., Huang, Q. (2018). Efficient multispike learning for spiking neural networks using probability-modulated timing method. IEEE Transactions on Neural Networks and Learning Systems, 30(7): 1984-1997. https://doi.org/10.1109/TNNLS.2018.2875471
[30] Zhou, S.B., Li, X.H. (2021). Spiking neural networks with single-spike temporal-coded neurons for network intrusion detection. arXiv preprint arXiv:2010.07803. https://doi.org/10.48550/arXiv.2010.07803
[31] Xu, Y., Yang, J., Zhong, S. (2017). An online supervised learning method based on gradient descent for spiking neurons. Neural Networks, 93: 7-20. https://doi.org/10.1016/j.neunet.2017.04.010
[32] Thiruvarudchelvan, V., Crane, J.W., Bossomaier, T. (2013). Analysis of SpikeProp convergence with alternative spike response functions. In 2013 IEEE Symposium on Foundations of Computational Intelligence, Singapore, pp. 98-105. https://doi.org/10.1109/FOCI.2013.6602461
[33] Lantz, B., Heller, B., McKeown, N. (2010, October). A

network in a laptop: Rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, New York, United States, pp. 1-6. https://doi.org/10.1145/1868447.1868466

[34] Kushi, K.S., Enayet, T., Dipa, O.A. (2019). Software-defined networking and its implementation by using Mininet. Doctoral dissertation. East West University.

[35] Patil, P.B., Bhagat, K.S., Kirange, D.K., Patil, S.D. (2020) Software defined networks using Mininet. International Journal of Recent Technology and Engineering 9(1): 843-849. https://doi.org/10.35940/ijrte.F9375.059120

[36] Al-Azawee, S.J., Al-Jamali, N.A.S. (2025). Handling heterogeneous traffic for software defined data-center network using spike neural network. Journal of Engineering, 31(5): 21-34. https://doi.org/10.31026/j.eng.2025.05.02