Computers, Materials & Continua
DOI: 10.32604/cmc.2023.031719
Article

Tech Science Press

Detection Collision Flows in SDN Based 5G Using Machine Learning Algorithms

Aqsa Aqdus[1], Rashid Amin[1,*], Sadia Ramzan[1], Sultan S. Alshamrani[2], Abdullah Alshehri[3] and El-Sayed M. El-kenawy[4]

[1]Department of Computer Science University of Engineering and Technology, Taxila, 47050, Pakistan
[2]Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, Taif, 21944, Saudi Arabia
[3]Department of Information Technology, Al Baha University, Al Baha, Saudi Arabia
[4]Department of Communications and Electronics, Delta Higher Institute of Engineering and Technology, Mansoura, 35111, Egypt
*Corresponding Author: Rashid Amin. Email: rashid4nw@gmail.com
Received: 25 April 2022; Accepted: 17 June 2022

Abstract: The rapid advancement of wireless communication is forming a hyper-connected 5G network in which billions of linked devices generate massive amounts of data. The traffic control and data forwarding functions are decoupled in software-defined networking (SDN) and allow the network to be programmable. Each switch in SDN keeps track of forwarding information in a flow table. The SDN switches must search the flow table for the flow rules that match the packets to handle the incoming packets. Due to the obvious vast quantity of data in data centres, the capacity of the flow table restricts the data plane's forwarding capabilities. So, the SDN must handle traffic from across the whole network. The flow table depends on Ternary Content Addressable Memorable Memory (TCAM) for storing and a quick search of regulations; it is restricted in capacity owing to its elevated cost and energy consumption. Whenever the flow table is abused and overflowing, the usual regulations cannot be executed quickly. In this case, we consider low-rate flow table overflowing that causes collision flow rules to be installed and consumes excessive existing flow table capacity by delivering packets that don't fit the flow table at a low rate. This study introduces machine learning techniques for detecting and categorizing low-rate collision flows table in SDN, using Feed Forward Neural Network (FFNN), K-Means, and Decision Tree (DT). We generate two network topologies, Fat Tree and Simple Tree Topologies, with the Mininet simulator and coupled to the OpenDayLight (ODL) controller. The efficiency and efficacy of the suggested algorithms are assessed using several assessment indicators such as success rate query, propagation delay, overall dropped packets, energy consumption, bandwidth usage, latency rate, and throughput. The findings showed that the suggested technique to tackle the flow table congestion problem minimizes the number of flows while retaining the statistical consistency of the 5G network. By putting the proposed flow method and checking whether a packet may move

from point A to point B without breaking certain regulations, the evaluation tool examines every flow against a set of criteria. The FFNN with DT and K-means algorithms obtain accuracies of 96.29% and 97.51%, respectively, in the identification of collision flows, according to the experimental outcome when associated with existing methods from the literature.

Keywords: 5G networks; software-defined networking (SDN); OpenFlow; load balancing; machine learning (ML); feed forward neural network (FFNN); k-means; and decision tree (DT)

1 Introduction

Fifth-generation (5G) [1] networking is becoming one of the most active research topics and attracting interest. The broad adoption of 5G [2] technology, the continuous growth of cellular connections, and the convergence of new network techniques such as big data, the Internet of Things (IoT) [3], and cloud computing have supported the overall economic and social development. 5G [4] data centers network have emerged as a critical component of modern computing environments. 5G data centers network [5] will be implemented across numerous scattered geo-locations with the support of multi-tenant network layers and millions of resource nodes. Datacenter traffic is growing exponentially, and the need for network capacity is increasing. Many novel data center network topologies, such as Monsoon [6], fat-tree, soon [7], Helios [8], and BCube [9], have been deployed to address the need for network capacity for data center internal communication.

SDN [10] is a new software-based network architecture and technology that allows for centralized state control. The OpenFlow [11] protocol has evolved into a critical component of SDN [12] development. The controller forms a control plane, while the switch performs convection routing to build a data plane. The switch employs the OpenFlow [13] protocol to gather connection information across the network to regulate the arriving flows. As the growing technology to address the limits of 5G networks [14], the integration of 5G with SDN [15] is being observed. SDN [16] is a new network design that is often regarded as the primary answer to a variety of problems in older networks. The control plane is detached from the data plane in SDN [17]. This dissociation increases the 5G network throughput because nework [18] intelligence is centralized. Programmable functionalities should be moved from network devices to centralised application controllers. SDN [19] algorithms inform the network what to do in reaction to network modifications or dynamic flow pattern alterations. Numerous different device setups are eliminated due to central management. At scale, access points can be customized.

However, due to the difficulties of traffic diversification and the inappropriate design of the control and data planes, Quality of Service (QoS) [19] provisioning is the biggest issue in existing [20] networks. End-to-end QoS [21] support in existing network designs is a persistent issue. Although academic and commercial researchers have offered numerous solutions to the existing networking's QoS [22] restrictions, many have failed or never been adopted. In contrast to the constraints of conventional networking topologies, the SDN [22] concept has arisen. On the other hand, the traffic created by 5G [23] networks is heterogeneous. Video broadcasting, online gaming, cloud storage, and other classes demand dependable, consistent network services with rigorous QoS [24] standards. OpenFlow enables flow level automation in SDN [25], which may constantly programme the system according to client QoS needs and system traffic conditions. Interconnections are allotted to network traffic flows depending on their QoS needs. Each kind needs a particular amount of QoS [26], therefore various

flow rules and routing mechanisms must be used. Resource partitioning is also the most significant side effect of traffic flow. Assume there are two 12 Gb/s paths connecting nodes A and B. Each has 600 Mb/s of open bandwidth, and the flow from node A to node B is 900 Mb/s. The flow cannot be rerouted efficiently due to resource partitioning, yet the system has 1 Gb/s of free bandwidth capacity. Flow scheduling should be done from a global perspective of networking to avoid the consequences of resource partitioning, considering the influence on all other flows.

SDN [27] is a system design that aims to increase the QoS of existing networks while also providing a common interface for higher applications and terminal devices. Fig. 1 shows a typical SDN architecture to understand this technology better. The application, control, and data layers are the three main aspects of the SDN design, as shown in Fig. 1. Compared to traditional network design, the data and control planes are separated into two planes rather than exposed to the higher application plane. Because the control plane will handle the network hardware devices autonomously, this decoupling can allow the application to operate the network directly via a single interface provided by the control plane, rather than considering hardware upgrades as in a traditional network. Correspondingly, when network terminal devices try to deliver messages to applications, the SDN [28] architecture would provide a standard interface to solve the scaling issues caused by hardware differences.

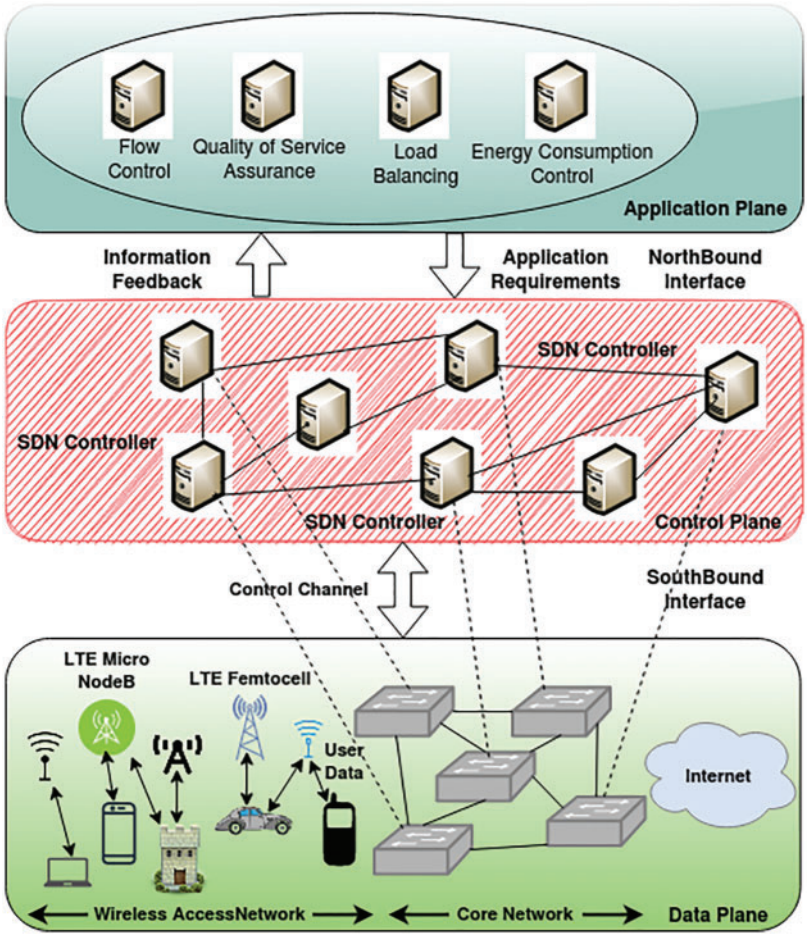

Figure 1: SDN-Based architecture on 5G Networks

However, the SDN framework comes with some flow vulnerabilities. Because the control plane handles a large amount of data from the whole network system, it must be fast. The flow table size limits the data plane's forwarding capabilities thus there is a congestion barrier. The flow table's storing and rapid searching of regulations relies on TCAM. The performance of TCAM is affected due to the size of data in the domain. TCAMs have high manufacturing costs and high-power consumption, limiting memory size. Currently, TCAM's several modern switches are carried hundreds of regulations, whereas data center traffic rates can reach huge numbers of flows per second, which necessitates the deployments and preservations of a wide range of flow regulations in TCAM memory. Traffic categorization is used in SDN networks to improve flow-table performance. The primary goal of traffic categorization is to provide the necessary degree of flow-table for incoming traffic. In order to identify such assaults in real-time, evict collision flows, and ensure the availability of flow tables and the efficiency of conventional flow rules, we confront the following challenges:

- Fast flow-table lookup while maintaining throughput of lookup engine, memory efficiency, and dynamic updates.
- What traits and approaches should the 5G network be employed to identify collision flows?
- Which features and techniques are chosen to detect collision and non-collision flows?
- How do follow regulations while transmitting packets when switches request the SDN controller?
- Flow-table updated dynamically during insertion and deletion process.

Consequently, these fundamental issues must be solved in SD-5G networks to increase flow table performance. The primary goal is to employ ML techniques to train and develop a useful and effective architecture for categorizing individuals with high detection accuracy. ML methods such as Feed Forward Neural Network (FFNN), K-Means, and Decision Tree (DT) for identifying and categorizing flow collisions in SDN are discussed in this work. The FFNN, K-Means, and DT algorithms were devised and implemented to enhance the efficiency of the flow collision recognition system. We've devised a solution to identify and prevent collision flows in real-time to address the challenges mentioned above. The key contributions or motivations of this paper are as follows:

- The QoS-aware optimization issue is mathematically formulated with minimum modifications to the forwarding tables.
- The resource partitioning issue in flow forwarding is being addressed.
- We've observed and defined new vulnerabilities induced due to the restricted flow table capabilities of OpenFlow switches.
- We use and implement machine learning algorithms, i.e., Feed Forward Neural Network (FFNN), K-Means, and Decision Tree (DT), that can successfully detect and categorize collision flows, including flow table capacity and flow table utilization in real-time.
- The suggested algorithm efficiency is evaluated using success rate query, propagation delay, overall dropped packets, the average energy consumption of 5G network, agent level hop to hop packet delay in 5G network, bandwidth usage, latency rate, and throughput metrics.
- To our knowledge, this is the first time that machine learning methods have been used to identify and categorize collision flows.

The remainder of this paper is arranged as follows. Section 2 presents an extensive analysis of the related work. In Section 3, the problem analysis and architecture design are described. In Section 4, the detail of the proposed model is discussed. In Section 5, the simulation parameters are described. In Section 6 experimental outcomes and performance analysis are discussed with the comparison of existing methods, and in Section 7 paper is summed up with the conclusion.

2 Related Work

Major past research efforts are evaluated to determine the research gaps. ML methods have opened hundreds of new possibilities for SDN deployment, notably in security-related applications. These methods are commonly used to improve SDN performance. Hamdan et al. [29] present a flow-concerned elephant flow recognition technique to solve traffic problems. The suggested solution used two classification techniques concurrently to achieve a reliable elephant flow identification: first on the SDN switches (i.e., switch-side classifier) and second on the controller (i.e., controller-side classifier). As a result, many mouse flows are examined in the switches, removing the requirement for the controller to process many classification requests and signaling alerts. Adedoyin et al. [30] introduce a complete overview of 5G enabling technologies for Uniform Distribution (UD) networks. The authors also describe research problems in sophisticated management strategies and backhaul solutions for 5G wireless networks, including needless Handover Optimization (HO), unequal radio resource sharing, power consumption, extreme interference, and diminished quality-of-service (QoS). The impact of different OpenFlow duration frames on the outcome estimation of different categorization techniques was examined in Khamaiseh et al. [31]. A total of 150 prototypes were constructed and evaluated using OpenFlow flow information collected in virtual and physical SDN conditions. The findings revealed that the OpenFlow traffic duration selected significantly influences recognition efficiency, with larger time windows that cause a lower detector outcome. Perera et al. [32] study shows how SDN services could use machine learning to identify network traffic. This strategy was extremely effective, demonstrating that this high-performance, intelligent-based transmission concepts may soon supplement or perhaps replace traditional network management.

Rasool et al. [33] suggested CyberPulse, a novel strong preventative monitoring approach that bases a machine learning-based classifier to minimize Loop-Free Alternate (LFA) in SDN. It organizes and classifies network traffic by using deep learning techniques. According to the findings, CyberPulse was able to identify abnormal flows and hence efficiently reduce them accurately. Dixit et al. [34] established a dynamic map of switches and controllers by concentrating on the fundamental design of distributed control plane. The authors create a controller pool, then assign it to different controllers based on controller consumption data. Chemeritskiy et al. [35] represent a QoS-based multi-path transmitting system for SDN. To fulfill the QoS needs of network services, this protocol creates a multi-path transmitting protocol based on SDN, which dramatically increases network utilization. Zhong et al. [36] present a scheme Smart Cooperative Platform for Load Balancing and Security (SCPLBS) for controller load balancing. The authors conducted trials by connecting three controllers C1, C2, and C3 to SCPLBS. In the function of Master, every single controller operates its area with varied network loads. The evaluation outcomes show a consistent CPU consumption under SCPLBS's load-balancing method. Wang et al. [37] present a load balancing method for link utilization based on the ant colony optimization algorithm Link Load-Balancing Algorithm based on Ant Colony Optimization (LLBACO). The control plane comprises four components: the monitoring component, the data gathering component, the load-balancing component, and the flows control component. This approach uses a dynamic threshold to pick the flow path and then saves them using the Ant Colony technique to determine the best path added to the best path list. Lan et al. [38] introduced two methods for dynamic path optimization: single-link and multi-link Dynamic Load-Balanced Path Optimization (DLPO) [39] techniques. The flow route comprising the majority of the top 10% heavily loaded links is then modified to a route where the blockage route comprises the most available bandwidth.

In conclusion, the research findings of routing methods in traditional wireless sensor networks have been plentiful. Research findings have also been obtained for software-defined routing protocols for future network designs. However, software-defined wireless sensor network technology is in its

infancy and needs to be researched and examined in various ways. So, it is essential to perform extensive study on its routing procedures.

3 Problem Analysis and Architecture Design

SDN-based data centers and many other SDN-based networking architectures are vulnerable due to the overflowing of flow-table. First the memory of the flow table is limited in size. The SDN switches search the flow table for the policies that match the incoming traffic to handle the incoming traffic. If no corresponding policy exists, the switches must send the packet on request to the control plane to seek metacognitive reading policies and establish new regulations. The overflow of the flow-table attacked on TCAM by sending some packets that were not suitable for the flow-table, causing new collision flow regulation for the flow-table and consuming free space. As a result, valid flow policies cannot be established and those policies that have been implemented are removed. The overflow of flow-table causes delays in processing and denial of service attacks that disrupt the overall network performance. The second issue is that, although we cannot offer any meaningful theoretical constraint on the maximal table occupancy if we observe the numerical pattern of traffic or table occupancy. Although if we knew that the table occupancy follow the normal distribution with defined means and standard deviations, we can't put a limit on how high it may go. Finally, the table occupancy distributions might have a high tail. The standard proactive technique in OpenFlow specification was investigated for flow table saturation.

We established the general operating 5G network model, described the unavoidable collision flows problems, and presented our solutions in this part. 5G networks are presently being established all over the globe. The design of this innovative network has significant needs. It brings major difficulties, such as managing enormous traffic growth, optimizing resource use, and providing high services and experiences. Because 5G networks offer real-time applications with massive data flows, they are extremely vulnerable to delay and latency, emphasizing the need for load-balancing as a user-centric networking strategy. Load balancing is necessary for 5G networks because they are created to attain a densified diverse network design with different RAN technologies. This demands the adoption of load balancing to optimize resource use, collision flows, imbalance load, controller failure, migration decisions and keep up with the enormous rise in traffic.

3.1 Problem Analysis

A connected graph G = (E, V) is widely used to illustrate a 5G network topology, where V is the collection of nodes and E is the collection of direct linkages among nodes. A route is usually indicated by limited separate nodes, such as $p = (v0, v1, v2, \ldots, vn)$, a.b. $\forall i \in [0, n], (vi, vi+1) \in E$. OSPF chooses this routing path in a typical manner. Furthermore, a 5G network is a significant surplus for other linkages. The major channel may be subjected to a data torrent that accelerates the development of multi-path route optimization systems. Now, we talk about the SDN perspective. The SDN comprises three equipment pairs: a controller K, a switch set A = a1, . . . , a|A|, and a host set B = b1, . . . , b|B|. The forwarding plane of an SDN is made up of these switches and hosts. The selection of routes is the responsibility of the controller. To put it another way, it will not take part in packet routing. So, graph G = (A B,F), in which F is a collection of links, may then be used to represent the network's topology. It's worth noting that one or more switches can only link any two terminals. Let k(f) and n(a) indicate the capacity of f link f F and the number of flow entries of a switch a A, respectively, for convenience of expression. Furthermore, let k(k) indicate the controller's capability. The controller's primary responsibility is to observe network traffic and issue Flow Entrance Installation Notifications

(FEIN). Because the needed resources km(k) for network observing are almost always stable and accessible, the capability of delivering FEIN is essentially constant. The standard controller may respond to flow-mod signals at a rate of 2 K per second. Whenever a collection of collision flows comes, we must install flow entries in the network for such flows $f = y_1, \ldots, y|f|$. Because every single flow may include a demand for specific resources, it is expected that the controller can obtain the traffic request r(y) for every $y \in f$. For example, a station traffic requirement is 4Mbps whenever it utilizes a high-definition video conferencing. For every flow, we utilize Py, which indicates the many possible pathways from source to destination. The definitions provide a thorough explanation of the issue.

- Offered a 5G network topology G = (E, V) and $\forall p \in$ Pi, j, the obtainable size *Cp* of route p is the lowest *ce* of all links in the route. $Cp = M\ ce, \forall e \in E$.
- Create a 5G network topology G = (E, V) and present link flow size *fe*, and the link bottleneck component is represented by *fe*/*ce*.
- Create a 5G network topology G = (E, V), a flow volume existing and the exist time σ of link e between all flow routes. Therefore, the capability restriction is $\forall p \in Pi, j, e \in E$, $\Sigma\sigma\ 1\ fp \approx Cp < ce$.

Fig. 2 shows upstream and downstream collision, which plays a crucial part in 5G networks, certain connections of a channel attain maximum capacity. Still, others do not, resulting in low bandwidth utilization were S6 S7 and S8 experience congestion on the upstream connection, whereas S3 and S4 experience congestion on the downstream connection.

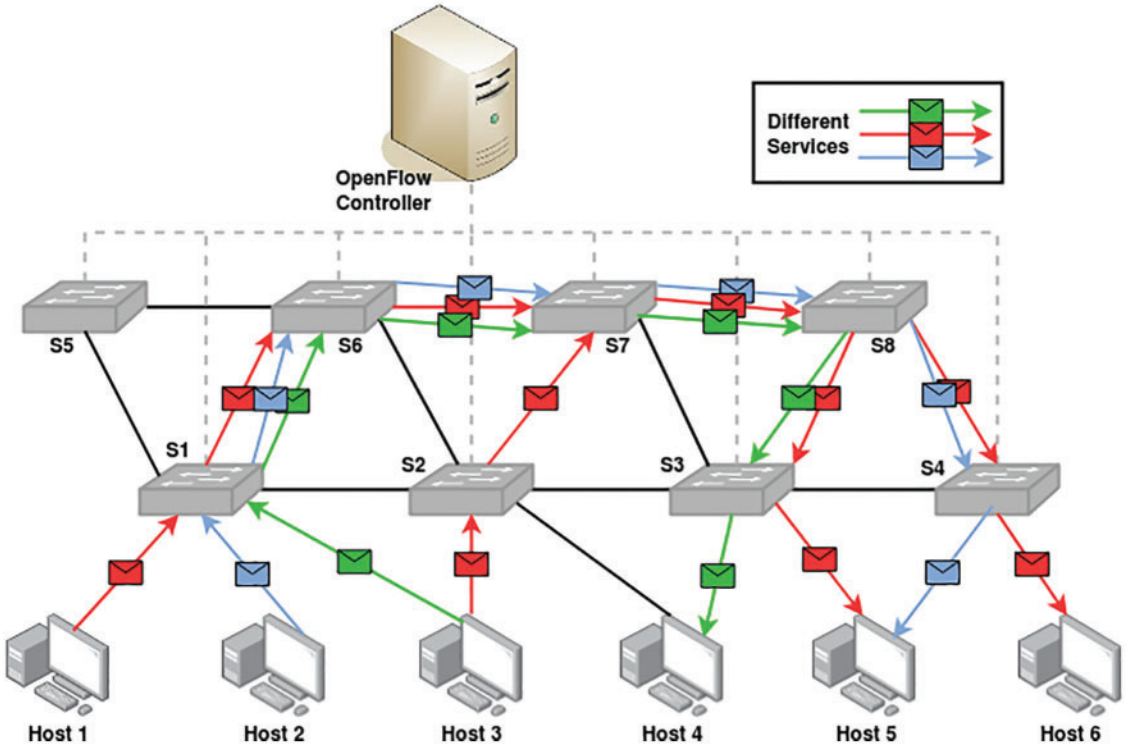

Figure 2: Problem analysis for an upstream and downstream traffic collision

3.2 Architecture Design

Data clustering is a critical and commonly used approach in data analysis and data mining. The main goal of clustering is to divide a dataset element into subsets, where elements of the same group are more similar. Several clustering algorithms are available in machine learning, one of which is K-Means clustering, a simple and efficient strategy. It is quick and simple to apply. However, the suggested system uses K-Means to construct task clusters to decrease make-span and provide a fair workload allocation across the 5G network. The different data services set is the dataset, consisting of a batch of tasks allocated to resources. The gap between task durations is used to group tasks. The number of clusters to be determined before using K-Means is denoted by the letter K. The procedure begins by selecting a centroid value for each cluster. Task lengths are the data points in the proposed solution since the tasks need to be grouped. Based on their processing capabilities, tasks are grouped and scheduled for 5G networks. As a result, the 5G network has a balanced workload.

We want to communicate our idea through Fig. 3, as indicated by the above mentioned issues. The system architecture is denoted by G (E, V), where E is the number of switches and V denotes the number of physical connections between switches. Furthermore, |M| denotes the number of switches, whereas |V| denotes the range of physical connections. The controller is denoted by K, furthermore, KN is used to indicate the connection between the location of the controller and the location of the switches. The variety of sub is indicated by C, and the semi-network is denoted as $S(E_i,V_i)$, where E_i denotes the set of switches in the semi-network and V_i is the set of physical connections in the semi-network. The semi-network controller is designated as K_i.The flows have been correctly timed to prevent network congestion. We identify the problem and offer an Open Flow-based load balancing solution to address the issues while also improving traffic supervising. There are two advantages that we can obtain. First is the identification of intelligent flow, and the second is clustering. The base of the problem, as previously stated, flows collision. We need to tailor measures to individual flow circumstances to minimize bandwidth congestions based on flow categorization.

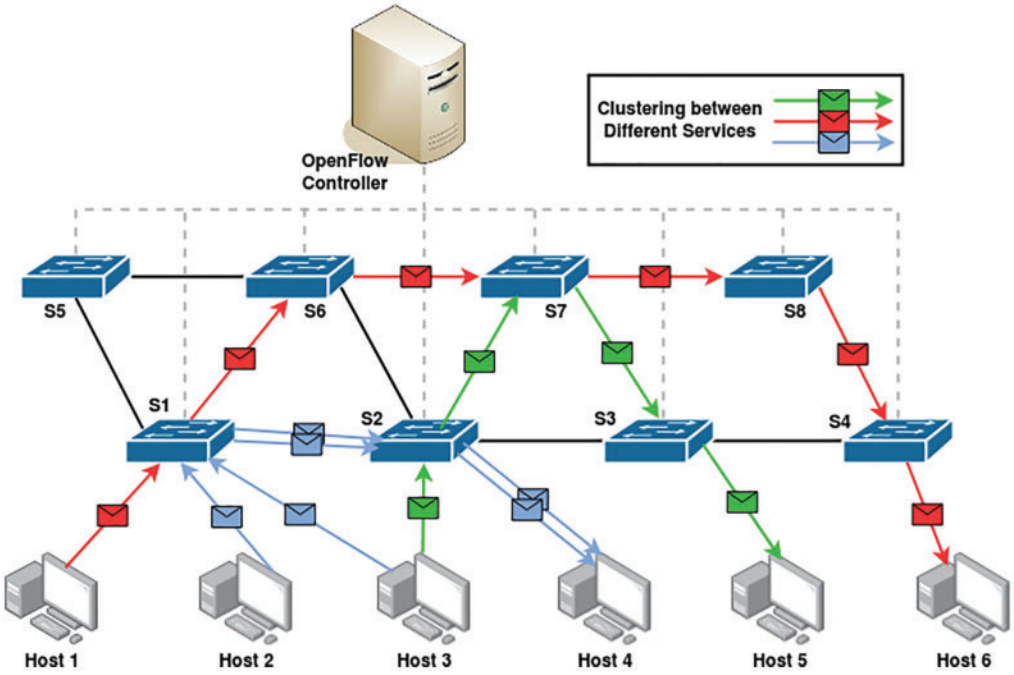

Figure 3: Anticipated outcomes for load balancing solution

4 Proposed Methods

In this section, we present a fat-tree and binary tree structure that focuses on memory capacity and efficiency of flow-table. The detection and classification stages are the two key phases of the proposed solution in this research. The suggested methodology for detecting and clustering collision flows is shown in Fig. 4. The first step is to distinguish between collision and non-collision flows. At first, the created flows are examined through the algorithms implemented on the control plane to monitor the behavior of current flows. Elements in the current flows, such as Mac addresses, IP addresses, and actions, distinguish between collision and non-collision flows. As a result, the results of these feature checking can determine whether the flows are normal or conflicting. Non-collision flows are delivered immediately to "OpenFlow," but conflicting flows are routed to the next step for clustering into the form of collision that occurs in conflicting flows. In "OpenFlow" two techniques for detecting collision flows have been suggested, Feed Forward Neural Network (FFNN) and Decision Tree (DT) techniques. The suggested architecture second phase is the clustering of collision flows. In this stage, an algorithm is developed in the controller, which checks the collision flows found in the detection process to identify flow patterns. Priority, IP address, and action are the three main characteristics of collision flows. The collision kinds are divided into seven types when the verification procedure is completed: Overloading (over), redundancy(redun), correlation (corre) A, correlation (corre) B, shadowing (shad), imbrication (imbri), and generalization (gene) showing in Algorithm 1 and Algorithm 2. To prepare flows gathered from the OpenFlow switch for the detection procedure, the pre-processing model is presented and deployed. The main characteristics (activity, protocol, IP address, and mac address) selected for learning the algorithms were retrieved from the flows. The activity, protocol, IP address, and mac address of the flow rule may all be used to specify and classify the sort of collision. According to the flow rule in the open flow switch, flows shall be considered collision flow entry.

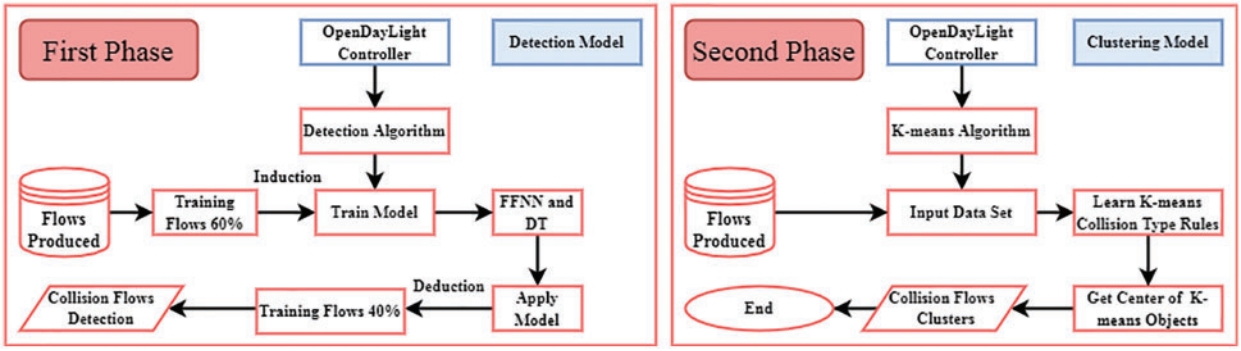

Figure 4: The proposed architecture for the detection and clustering of collision flows

4.1 Feed Forward Neural Network (FFNN) with Decision Tree (DT) Algorithm

The Feed-Forward Neural Network (FFNN) and Decision Tree (DT) methods are the machine learning algorithms used to solve regression and categorization issues. A multi-layer FFNN design adopte consisted of a multi-layer feed-forward network with a sigmoid hidden and linear output layer. The network can be trained through FFNN non-linear and linear correlations among input and output matrices. The linear output layer enables the network to generate values with –1 to +1 biases. Provided stable input and enough neurons in its buried layer, this network can handle multi-dimensional mappings tasks arbitrarily well. This architecture primarily consists of a FFNN multi-layer architecture with straight strokes as parameters. In our case, the architecture comprises an

input layer, a hidden layer with some neurons, and an output layer. The FFNN output layer is not connected to the hidden layer (for instance, the first layer of a two-layer feed-forward network). DT is a tree-shaped technique in which inner nodes and stems specify database features that reflect decision procedures, with each leaf node indicating the results. Resultant nodes are used to make a decision, and they also include many branches, which results in leaf nodes with no more stems. Particularly the leaf node in this approach is capable to identify the flow. The schematic depiction of the overall formation of the DT technique is shown in Fig. 5. Decision trees may alternatively be described as a combination of mathematical and analytical approaches to identify, categorize, and generalize a particular data set. Data is stored in the manner of records, as illustrated in the Eq. (1):

$$(a, B) = (a_1,\ a_2,\ a_3, \ldots\ a_n,\ B) \tag{1}$$

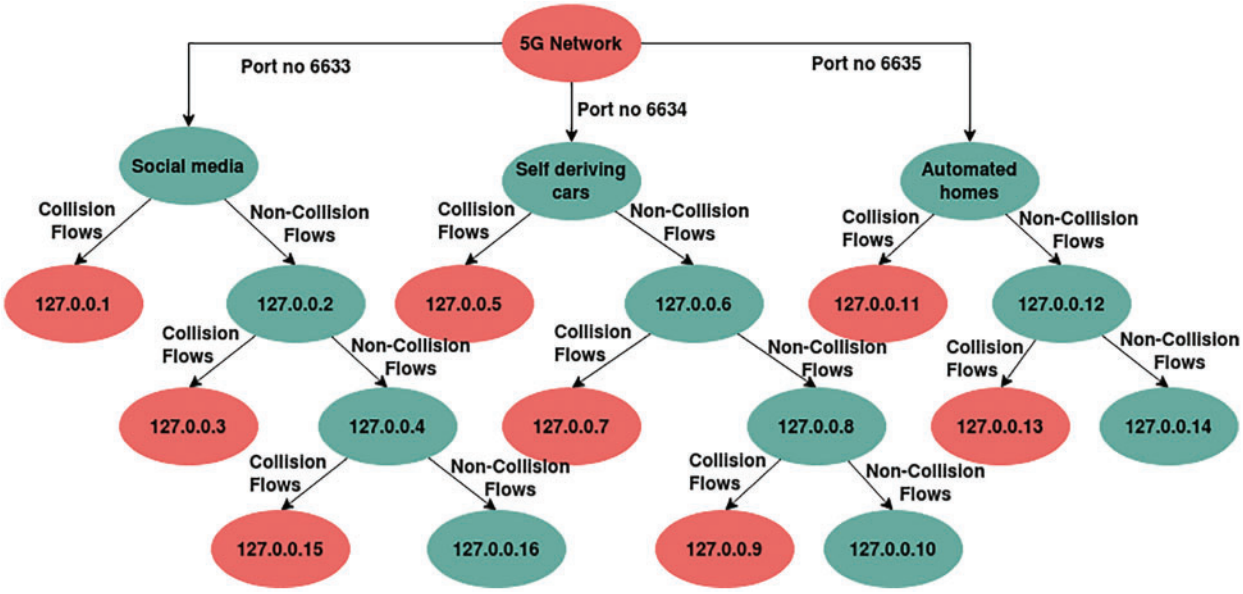

Figure 5: The diagram of FFNN with DT algorithm

The conditional factor B is the target variable that learning seeks to identify or categorize. The vector a is made up of the learning qualities a_1, a_2, a_3, etc. In addition, the implementation phases of FFNN with DT as shown in Algorithm 1 for the detection of collision flows. The control plane implements decision tree elements and configures the learner feature. Start preparing and acquiring all created flows from the OpenFlow switch for all flow sizes. Use 60% of the created flows to train the models. Evaluate the algorithm's efficiency by determining the outcome of the remainder 40% of created flows. Calculate the time complexity for the DT method by evaluating the discriminant running. The FFNN and DT algorithms are implemented to increase the speed and accuracy of execution. The pseudo-code for algorithms employed in identifying collision flows is shown in Algorithm 1. In addition, the following processes are involved in the detection process: Algorithm's implementation and execution for detection of collision flows. The proposed model distinguishes between collision and non-collision flows. Non-collision flows push to OpenFlow table. The collision flows are sent towards the clustering algorithm.

4.2 K-Means Clustering Algorithm

K-Means clustering is a part of unsupervised learning. The main purpose of this technique is to discover clusters in datasets, and K denotes the number of groups. Different classes use a decision border in the categorisation technique shown in Fig. 6. The K-Means technique begins with preliminary K centroids drawn randomly from the dataset. The method alternates between allocating data points and updating centroids in two stages. The data point is allocated to its closest centroid using the squared Euclidean distance in this phase. It is relatively efficient, and its computational complexity is O(nkt). Where 'n' indicates the number of occurrences, k indicates the number of clusters and t indicates the number of repetitions. The second step of the suggested model is the categorization of collision flows. In this stage, an algorithm is applied in the control plane, checking the collision flows in the detection phase to examine flow action. Priority, action, and IP address are the three main characteristics of collision flows. When the verification procedure is completed, the collision flows are divided into seven types. Overloading (over), redundancy(redun), correlation (corre) A, correlation (corre) B, shadowing (shad), imbrication (imbri), and generalization (gene). The pseudo-code for the K-Means method categorizes collision flows is shown in Algorithm 2.

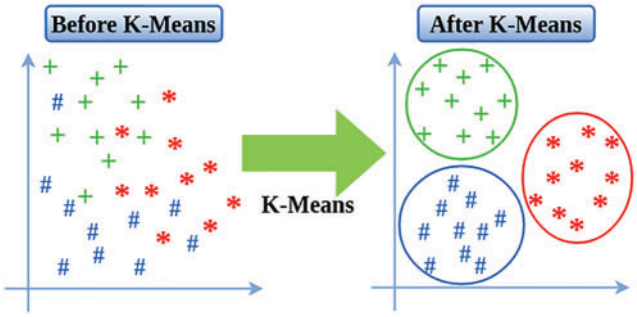

Figure 6: Different distinct groups for the K-Means algorithm

The following are the processes concerned in the categorization phase: Firstly, implementation & execution of K-Means categorization algorithm. Secondly, flow detection is started by the proposed algorithm. Thirdly, the proposed technique examines the flows' priority and IP address information. At last, the system categorization of collision flows based on the features identified in step 3. K-means approach is also called K∗-means used to produce good clustering results for computation resource and energy consumption. To improve the chance clustering result for computation resource and energy consumption, containing one or more beginning points, K-means expand the amount of initial clustering centers. We will acquire more than K clusters if we employ more than K starting centers. K-means employ a technique known as Top-n closest cluster merging to limit the number of clusters. Although K-means can improve clustering. Rather than Euclidean distance, we utilize the length of the physical connection and stipulate that the beginning spots must be picked from the position of switches. Therefore, we can only obtain a method that reduces the maximum propagation delay between the switches and their controller. We offer a mechanism to substitute the shortest length of physical connection, which we name the technique after substituting the Top-n appropriate cluster merging, in terms of addressing both computation resource and energy consumption. The following is the function in the Eq. (2):

$$f\left(N_i,\ N_j\right) = \partial \frac{B_e * C_{ij}}{\Sigma_{I=1}^{Be} C_{ij}} + (1-\partial)\ \frac{B * (|C_i|+|C_j|)}{|N|} \tag{2}$$

Algorithm 1: The pseudo-code for the detection of collision and non-collision flows

Input: Flow_1 ó, Flow_2 ń, ó addr, ń addr, Priority B, Protocol C, Action Ã.
Output: Collision Flows
Process: Collision Flows Detection ()

```
1. while Có = Cń                                          \\   Non collision flow
2.   while Ãó = Ãń
3.         while Bó > Bń
4.               while ó.addr ⊆ ń.addr then or ń.addr ⊆ ó.addr
5.                   return collision flow                \\ Redundancy collision flow
6.               endwhile
7.            else
8.               while Bó = Bń
9.                   while ó.addr = ϕ oor ń.addr = ϕ
10.                      return collision flow            \\ Imbrication collision flow
11.                  endwhile
12.              else
13.                  while Bó < Bń
14.                      while ó.addr ∩ ń.addr
15.                          return collision flow        \\ Shadowing collision flow
16.                      endwhile
17.                  endwhile
18.              endwhile
19.        endwhile
20.        else
21.           while Ãó ≠ Ãń
22.              while Bó < Bń
23.                 while ń.addr ⊆ ó.addr then or ó.addr ⊆ ń.addr
24.                       return collision flow          \\ Generalization collision flow
25.                    else
26.                       while ó.addr ∩ ń.addr
27.                           return collision flow      \\ Correlation collision flow
28.                       endwhile
29.                   endwhile
30.                else
31.                   while Bó = Bń
32.                   while ń.addr ⊆ ó.addr then or ó.addr ⊆ ń.addr
33.                       return collision flow          \\ Overlapping collision flow
34.                   endwhile
35.                endwhile
36.             endwhile
37.          endwhile
38.     endwhile
39. endwhile
```

Algorithm 2: The pseudo-code for the categorization of collision and non-collision flows

Input: Flow_1 ó, Flow_2 ń, ó addr, ń addr, Priority B, Action Ã.
Output: over, redun, corre A, shad, corre B, imbri, and gene.
Process: Collision Flows Categorization ()

```
1.  while Bó = Bń
2.      while Bó > Bń
3.          return redun collision          \\  Cluster of redundancy collision flow
4.        else
5.          while ó.addr ∩ ń.addr
6.              while Ãó = Ãń
7.                  return over collision   \\  Cluster of overlapping collision flow
8.                else
9.                  return corre (A) collision  \\  Cluster of correlation (a) collision flow
10.             endwhile
11.           else
12.               while ó.addr = ń.addr
13.                     return shad collision   \\  Cluster of shadowing collision flow
14.               else
15.                   return gene collision     \\  Cluster of generalization collision flow
16.           endwhile
17.         endwhile
18.     endwhile
19.       else
20.         while ó.addr = ϕ or ń.addr = ϕ
21.           return imbri collision            \\  Cluster of imbrication collision flow
22.         else
23.             return corre (B) collision      \\  Cluster of correlation (b) collision flow
24.         endwhile
25. endwhile
```

The description of function are as follows:

- ∂ is a variable that ranges from 0 to 1. So when a component is increased, it focuses so much on computation resources, and when it is decreased, it focuses so much on energy consumption.
- The method's initial section is concerned with computation resources. B_e denotes the latest number of clusters, whereas C_{ij} denotes the smallest physical path among clusters i and j.
- The energy consumption portion of the function is the second version. B denotes the desired number of clusters, whereas |Ci| denotes the number of switches in cluster i.

After the explanation of function, there are the following working steps in the K-Means clustering algorithm:

- Select the amount of clusters K. Choose any K data points randomly as cluster centers. Choose cluster centers that are as far apart as feasible from one another.
- Determine the Euclidean distance and the center of every cluster. The distance can be determined using the provided Euclidean distance or the matrix formula.

- Every data point should be assigned to one of the clusters. A data point is allocated to the cluster with the center that is nearest to it. Calculate the center of freshly generated clusters once again. All the data points in a cluster are to calculate the cluster's center.
- Continue the operation from step 2 through 3 until one of the following halting conditions is met: The center of freshly created clusters remains constant, in the same cluster, data points are still present, and the total number of iterations has been achieved.

Additionally, the K-Means algorithm execution phases can be summed in the following manners: Add K-Means components in the OpenDayLight (ODL) controller. Configure the linear module's learner and use the hard margin option. Compile and acquire all created flows from the OpenFlow switch for all flow sizes. Use 60% of the produced flows to train the K-Means classifier. Evaluate the classifier's efficiency by estimating the outcome for the last 40% of created flows. Calculate the time complexity for the K-Means method by evaluating the discriminant matrix.

Fig. 7 shows the proposed model's flow diagram, which is utilized to create and make the flows. The proposed model consists of two steps detection and categorization using a flow range of 5000 to 15000 with a 1000 flow increment each step in two network topologies, Fat Tree and Simple Tree Topologies, generated with the Mininet simulator and coupled to the OpenDayLight (ODL) controller. Numerous flows have been created using the src/dst ips, src/dst ports, and communications protocol. The controller receives every packet, starting a newfound flow in the switch. Normally, afterward, the creation of topologies with the topo application, the quantity of flows is defined, and the ODL supervisor application is launched to produce regular flows. The collision regulations are applied in the ODL controller by executing the collision flow application after the specified number of flows have been created. When all the collision and non-collision flows have been created, the flow stat app gathers and stores all the flows into a CSV file. To examine the efficiency of the suggested methods the detection and categorization of collision flows in perspective of proficiency and effectiveness. We used several assessment criteria: success rate query, propagation delay, dropped packets, average energy consumption, agent level hop to hop packet delay, bandwidth usage, latency rate, and throughput.

5 Simulation Parameters

We used a simulation environment with well-known networking tools and devices to evaluate the proposed system's performance and efficiency based K-Means. The Ubuntu 20.04 LTS operating system, Mininet simulator (a software application that simulates a network on a laptop), OpenFlow switches and routers, topology generator, and other tools are included in the used simulation environment. In the beginning, a topology is developed with this tool and then merged with the SDN controller. Then, in Mininet, we add many SDN switches, hosts, and a controller to form a network corresponding to our issue description and solution. We tested 35, 45, 55, 65, 70, 75, 80, and 85 nodes in Mininet v 2.3.0 with 5G network services requirements transmission range. The random waypoint model follows the nodes. We begin by looking at the effect of ∂ computation resources and energy consumption. The efficiency of our method is then assessed by comparing it to existing techniques. We utilize a scale called Software Defined-Networking Standard Deviation (SDNSD) to evaluate the efficacy of resource allocation, which is the sum of the standard deviation of the number of switches handled by the controller. The calculation technique of SDNSD is shown in Eq. (3), where C is the number of switches inside the network, $|M|$ is the total amount of switches, and $|M_i|$ is the number of

switches inside the network. The lower value of SDNSD, as can be observed from the formula.

$$SDNSD = \sqrt{\frac{1}{C}\Sigma_{i=1}^{C}\left(|M_I| - \frac{|M|^2}{C}\right)} \tag{3}$$

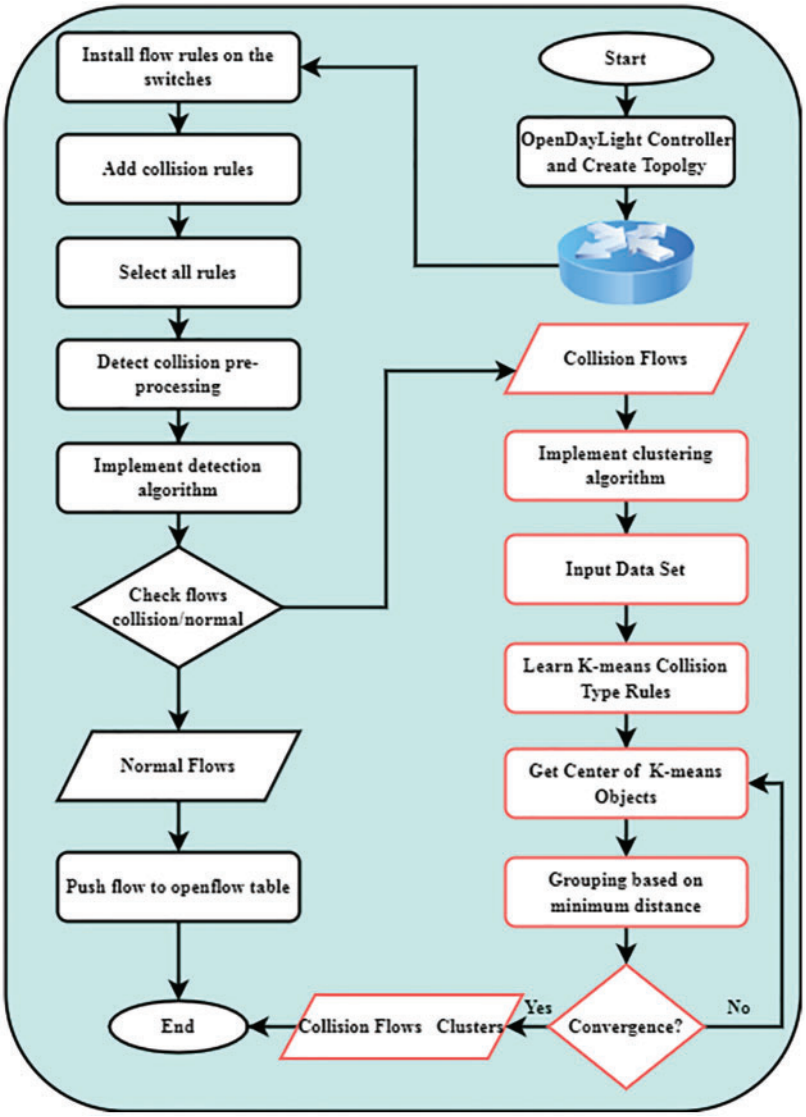

Figure 7: Flow diagram of the proposed solution

Continuing the explanation of the SDNSD parameter, we compute the amount of C∗. As a consequence, throughout our research, we fixed C∗ to 2C. The control parameter approach is used to investigate the influence of ∂ on computation resources and energy consumption. Because we set C to 6, the value of C∗ equals 12. In addition. By putting the size of ∂ from 0 to 1 with different increments, we can determine the value of SDNSD and the maximum advantage. We do not specify a method for gathering diagnostic characteristics for detecting and categorising collision flows. Therefore the proposed method was evaluated using the NLS-KDD dataset SDN based which is publiclay available.

We select different machine learning techniques, i.e., Fuzzy, SVM, and Hac, to compare our proposed solution. Finally, we look at success rate query, propagation delay, overall dropped packets, the average energy consumption of 5G network, agent level hop to hop packet delay in 5G network, bandwidth usage, latency rate, and throughput. We implement it to identify and analyze the effectiveness of the suggested detection and categorization model based on SDN switches. To assess the system's malicious attack efficacy, we first identify whether the attack mitigation module's flow rules are attack flows. Every second, we gather the contents of the flow table of the affected switch without a proposed model and determine the proportion of attack flows in it. When the network is under assault for the first 100 s and the system is not started, the percentage of attack flows increases until it reaches 60% of the entire flow table capacity. The fraction of assault flows reduced significantly once the proposed model was triggered at the 100th second, remaining below 12% of the flow table capacity. It means the system can precisely detect attack flows in the flow table and evict them while allowing legitimate flows to be installed and sent.

Flow rules acquired from the OpenFlow switch for the detection process are introduced and implemented to adapt. The flow rules were collected to show the key properties (activity, IP address, protocol, and mac addresses) selected for execution of the algorithm. The interconnection between machine learning outputs and SDN control cannot cause overhead latency. Because when load exceeds the capacity of the switch, the proposed method for the SDN controller shifts the load to other switches to avoid the congestion. The simulation parameters are described in Tab. 1.

Table 1: Simulation parameters

Simulation parameters	Values
Simulation area	1200 × 1200
Simulator	Mininet-wifi
Number of switches and controllers	20, 1
Type of SDN controller	OpenDayLight (ODL)
Number of SDN domain	5
Simulation time	100 s
Traffic type	UDP and TCP
Flow table size	1200 entries
Flow size	120 bytes
Number of packets	100
Packet size	512 bytes

6 Experimental Results and Metrics

The following measurement system is used to validate the efficiency of the suggested approaches, which are currently under debate. (1) success rate query (2) propagation delay (3) overall dropped packets (4) average energy consumption of 5G network, (5) agent level hop to hop packet delay in 5G network, (6) bandwidth usage (7) latency rate (8) throughput. The following metrics are defined as follows:

6.1 Success Rate Query

The success rate query is an essential factor for determining the effectiveness of the network source approach. It's a fraction of the total number of demands made to the total number of successful responses. The following Eq. (4) is how it is stated as a percentage:

$$\text{Success rate query} = \text{Repl}/\textit{Reque}^{*100} \tag{4}$$

Here, Reque denotes the number of resource demands sent out and Repl denotes the number of appropriately received resource replies. Fig. 8 depicts the successful requests as a fraction of the total created queries to successful responses during the simulation. Our proposed technique shows better performance as compared to traditional methods. The traditional methods are ineffective as shown in Fig. 8.

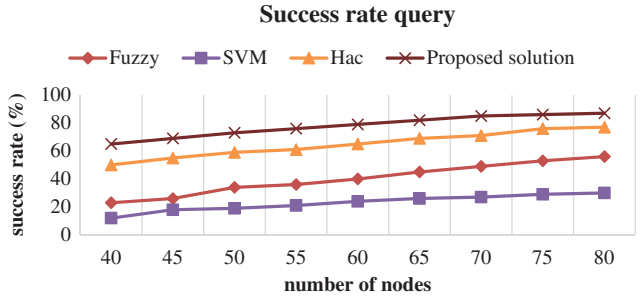

Figure 8: Comparison of success rate query with the existing techniques

6.2 Propagation Delay

The time it needs to go from one end of a connection to another is propagation delay. The delay is determined by the distance (DDD) between the transmitter and receiver and the wave signal's propagation speed (SSS). The following Eq. (5) calculates it:

$$P_d = \text{distance}/\text{propagation speed (in ms)} \tag{5}$$

Here, Distance: When the distance between the medium and the destination is greater, it takes longer to reach the goal.

Propagation speed: The packet will be received more quickly if the signals of propagation speed are higher.

Fig. 9 depicts the propagation delay of several nodes with various thread counts. Compared to the other techniques, the results indicate that the proposed method performs much better and the performance remains the same when traffic increases or decreases.

6.3 Overall Dropped Packets

Small chunks of information known as packets are transmitted and taken while using cyberspace or any connection. Packet drop happens when more than one packets struggle to meet their desired target. Packet loss inductions cause network disruption, delayed maintenance, and even the extensive loss of network access for users. Packet loss can influence any service, but services that depend on real-time packet processing, such as video, audio, and gaming systems, are the most frequent targets. Packet

loss can be influenced by a lot of aspects, the most frequent of which include network congestion, software defects, network hardware failures, and security risks. Fig. 10 summarizes the output of the dropped packets. The discrepancy between data packets sent and received is used to calculate the total dropped packets. The complete number of packets was lost throughout the simulation process. The graph indicates that our recommended technique performs better than traditional methods and does not affect the resource discovery process under different nodes.

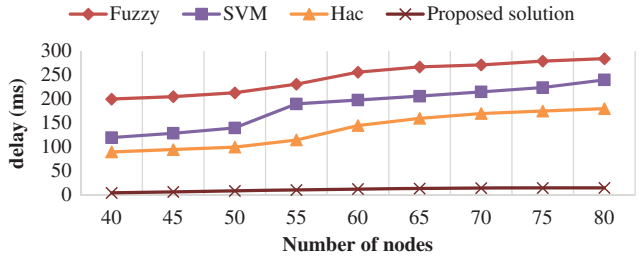

Figure 9: Comparison of propagation delay with the existing techniques

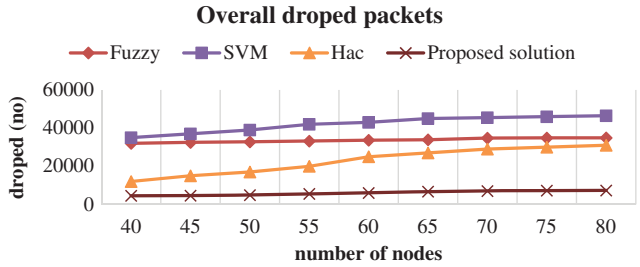

Figure 10: Comparison of dropped packets with the existing techniques

6.4 Average Energy Consumption of 5G Network

The spent energy is generally determined as the total energy expended by all the network's nodes during the simulation's duration. Joules are the units of measurement. The formula is as follows in Eq. (6) :

$$X_{avg} = 1\backslash \mathrm{N}\ \Sigma_{ni} = 1B_{ai} - C_{ai}\ \text{(in joules)} \tag{6}$$

Here,

Aavg: The computed avg spent energy of 5G network in Joules.

N: The complete number of nodes in the 5G network.

BAi: The preliminary battery energy is denoted by the letter i.

CAi: The final battery energy of the node i.

Fig. 11 shows how different methods perform in respect of energy use. Because the network nodes in Mininet are battery-powered, power conservation is critical. The graph shows that our suggested technique uses less battery power than current techniques. The SVM procedure is ineffective.

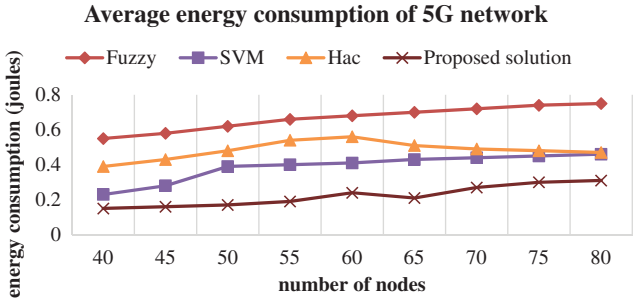

Figure 11: Comparison of average energy consumption with the existing techniques

6.5 Agent Level Hop to Hop Packet Delay in 5G Network

This measurement system is a more sophisticated variant of the standard delay computation. Rather than computing the delay among source and destination. The delay is computed for every packet created throughout the search process. During the forwarding of the resource discovery request, it is the time it takes for a packet to go every hop. The formula is as observes in Eq. (7):

$$\text{Hop to Hop Packet Delay} = T_a - T \quad \text{(in ms)} \tag{7}$$

Here,

Ta: is the moment when a node gets a request for source discovery.

Tb is the time that the preceding hop forwarded the resource discovery request.

Fig. 12 shows how hop-based packet delay is used to analyze the 5G network protocol's efficiency. For every resource discovery procedure, only successfully transmitted data packets are examined. The SVM stroll fails miserably. The findings of our proposed approach are promising.

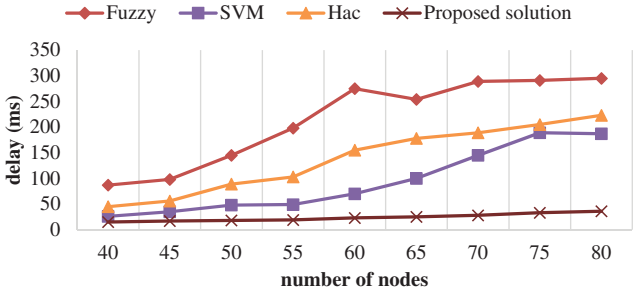

Figure 12: Comparison of hop-to-hop packet delay with the existing techniques

6.6 Bandwidth Usage

A bandwidth usage verifies how much data it can deliver and collect simultaneously. The amount of water flowing across a channel may be associated with bandwidth in concept. More water can flow through a pipe with a larger diameter. The concept of bandwidth seems to be the same. The higher capacity means that more data can pass over a communication channel per second. The bandwidth usage is utilized in Fig. 13 to verify the proposed solution efficiency. The simulation results are highly

accurate. Under various nodes, the performance of the bandwidth system stays unchanged. The performance of our suggested technique surpasses existing techniques.

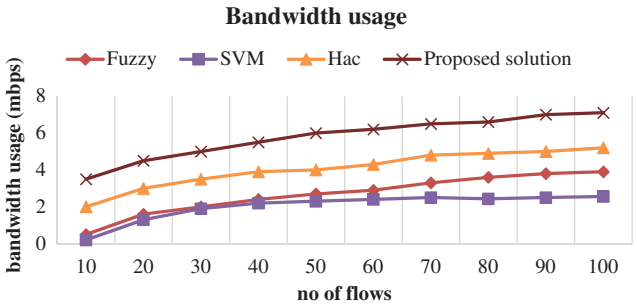

Figure 13: Comparison of bandwidth usage with the existing techniques

6.7 Latency Rate

The word "latency" indicates the time it needs for something to occur. It's commonly stated as a round trip delay or the time it uses for data to move from point A to point B. Because a computer utilizing a TCP/IP network delivers a finite volume of information to its target and then waits for an acknowledgement before transmitting any more, the round trip delay is essential. As a result, the round trip delay considerably influences network quality.

In many instances, latency is measured in milliseconds (ms). The findings demonstrate in Fig. 14 under a heavy load of no. of flows requests per second is reached, high load adds only a small amount of overhead. The controller handles the heavy number of requests per second, which equates to the different number of services transfer rate. The platform should handle even more requests if a greater number of flows are employed.

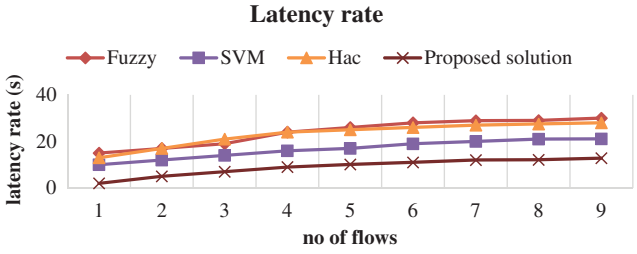

Figure 14: Comparison of latency rate with the existing techniques

7 Conclusions

In the SDN paradigm, this study chooses flow rule elements to detect malicious flows by analyzing the difference between normal and collision flows, using machine learning techniques to detect and categorize collision flows. The priority, protocol, action, and IP source address of flow rules are used to identify and categorize the forms of collision. This study used three algorithms to improve efficiency and efficacy: Feed Forward Neural Network (FFNN), K-Means, and Decision Tree (DT). The suggested algorithm's performance was assessed by utilizing evaluation measures such as success rate query, propagation delay, overall dropped packets, the average energy consumption of 5G network, etc. Our assessment reveals that the proposed method can impose varying aspects of consistency

validation in the context of flow updates and topology changes in an SDN network. We evaluated the double serial operation to the double parallel, expanding window, and sliding window methods using several views. The best results came from the double serial operation. According to the experimental outcome, the FFNN with DT and K-means algorithms obtain accuracies of 96.29% and 97.51%, respectively, in the case of collision flows detection and categorization. Thus, it is observed that the suggested methods obtained favorable results in identifying and categorizing collision flows in SDN. We also generalize this research in other networks (e.g., in wireless networks, IoT, data centers, cloud computing, distributed systems and many more others) besides 5G using different machine learning techniques.

Funding Statement: Taif University Researchers supporting Project number (TURSP-2020/215), Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study."

References

[1] X. Yuan, H. Yao, J. Wang, T. Mai and M. Guizani, "Artificial intelligence empowered QoS-oriented network association for next-generation mobile networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 856–870, 2021.

[2] A. Guo and C. Yuan, "Network intelligent control and traffic optimization based on SDN and artificial intelligence," *Electronics*, vol. 10, no. 6, pp. 700, 2021.

[3] T. G. Nguyen, T. V. Phan, D. T. Hoang, T. N. Nguyen and C. So-In, "Federated deep reinforcement learning for traffic monitoring in SDN-based IoT networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1048–1065, 2021.

[4] F. Paolucci, F. Cugini, P. Castoldi and T. Osiński, "Enhancing 5G SDN/NFV edge with P4 data plane programmability," *IEEE Network*, vol. 35, no. 3, pp. 154–160, 2021.

[5] Z. Dengyong, J. Hu, F. Li, X. Ding, A. K. Sangaiah *et al.,* "Small object detection via precise region-based fully convolutional networks," *Computers, Materials and Continua*, vol. 69, no. 2, pp. 1503–1517, 2021.

[6] S. Gonzalez-Diaz, R. Marks, E. Rojas, A. De La Oliva and R. Gazda, "Stateless flow-zone switching using software-defined addressing," *IEEE Access*, vol. 9, pp. 68343–68365, 2021.

[7] Y. Zhao, B. Yan, D. Liu, Y. He, D. Wang *et al.,* "SOON: Self-optimizing optical networks with machine learning," *Optics Express*, vol. 26, no. 22, pp. 28713–28726, 2018.

[8] M. N. Hall, K. -T. Foerster, S. Schmid and R. Durairajan, "A survey of reconfigurable optical networks," *Optical Switching and Networking*, vol. 41, pp. 100621, 2021.

[9] B. Mahapatra, A. K. Turuk and S. K. Patra, "Multi-tier delay-aware load balancing strategy for 5G HC-RAN architecture," *Computer Communications*, vol. 187, pp. 144–154, 2022.

[10] S. Prabakaran, R. Ramar, I. Hussain, B. P. Kavin, S. S. Alshamrani *et al.,* "Predicting attack pattern via machine learning by exploiting stateful firewall as virtual network function in an SDN network," *Sensors*, vol. 22, no. 3, pp. 709, 2022.

[11] S. Khan, A. Hussain, S. Nazir, F. Khan, A. Oad *et al.,* "Efficient and reliable hybrid deep learning-enabled model for congestion control in 5G/6G networks," *Computer Communications*, vol. 182, pp. 31–40, 2022.

[12] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez *et al.,* "A survey on machine learning techniques for routing optimization in SDN," *IEEE Access*, vol. 9, pp. 104582–104611, 2021.

[13] W. -K. Lai, Y. -C. Wang, Y. -C. Chen and Z. -T. Tsai, "TSSM: Time-sharing switch migration to balance loads of distributed SDN controllers," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1585-1597, 2022.

[14] D. P. Isravel, S. Silas and E. B. Rajsingh, "Centrality based congestion detection using reinforcement learning approach for traffic engineering in hybrid SDN," *Journal of Network and Systems Management*, vol. 30, no. 1, pp. 1–22, 2022.
[15] H. Zhong, J. Xu, J. Cui, X. Sun, C. Gu *et al.,* "Prediction-based dual-weight switch migration scheme for SDN load balancing," *Computer Networks*, vol. 205, pp. 108749, 2022.
[16] I. Leyva-Pupo and C. Cervelló-Pastor, "Efficient solutions to the placement and chaining problem of user plane functions in 5G networks," *Journal of Network and Computer Applications*, vol. 197, pp. 103269, 2022.
[17] Q. Yasin, Z. Iqbal, M. A. Khan, S. Kadry and Y. Nam, "Reliable multipath flow for link failure recovery in 5G networks using SDN paradigm," *Information Technology and Control*, vol. 51, no. 1, pp. 5–17, 2022.
[18] W. Wei, J. Yongbin, L. Yanhong, J. Li, X. Wang *et al.,* "An advanced deep residual dense network (DRDN) approach for image super-resolution," *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1592, 2019.
[19] P. D. Bojović, T. Malbašić, D. Vujošević, G. Martić and Z. Bojović, "Dynamic QoS management for a flexible 5G/6G network core: A step toward a higher programmability," *Sensors*, vol. 22, no. 8, pp. 2849, 2022.
[20] M. P. Nowak and P. Pecka, "Routing algorithms simulation for self-aware SDN," *Electronics*, vol. 11, no. 1, pp. 104, 2021.
[21] B. Mykola, H. Beshley, M. Medvetskyi, N. Kryvinska and L. Barolli, "Centralized QoS routing model for delay/loss sensitive flows at the SDN-IoT infrastructure," *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3727–3748, 2021.
[22] T. Prohim, S. Math, A. Lee and S. Kim, "Multi-agent deep Q-networks for efficient edge federated learning communications in software-defined IoT," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 3319–3335, 2021.
[23] B. Shariati, L. Velasco, J. -J. Pedreno-Manresa, A. Dochhan, R. Casellas *et al.,* "Demonstration of latency-aware 5G network slicing on optical metro networks," *Journal of Optical Communications Networking*, vol. 14, no. 1, pp. A81–A90, 2022.
[24] W. Jin, Y. Wu, S. He, P. K. Sharma, X. Yu *et al.,* "Lightweight single image super-resolution convolution neural network in portable device," *KSII Transactions on Internet Information Systems*, vol. 15, no. 11, pp. 4065–4083, 2021.
[25] M. Usman, R. Amin, H. Aldabbas and B. Alouffi, "Lightweight challenge-response authentication in SDN-based UAVs using elliptic curve cryptography," *Electronics*, vol. 11, no. 7, pp. 1026, 2022.
[26] M. U. Iqbal, E. A. Ansari, S. Akhtar and A. N. Khan, "Improving the QoS in 5G HetNets through cooperative Q-learning," *IEEE Access*, vol. 10, pp. 19654–19676, 2022.
[27] S. Ashtari, M. Abdollahi, M. Abolhasan, N. Shariati and J. Lipman, "Performance analysis of multi-hop routing protocols in SDN-based wireless networks," *Computers \& Electrical Engineering*, vol. 97, pp. 107393, 2022.
[28] R. Amin, N. Shah and W. Mehmood, "Enforcing optimal ACL policies using K-partite graph in hybrid SDN," *Electronics*, vol. 8, no. 6, pp. 604, 2019.
[29] M. Hamdan, B. Mohammed, U. Humayun, A. Abdelaziz, S. Khan *et al.,* "Flow-aware elephant flow detection for software-defined networks," *IEEE Access*, vol. 8, pp. 72585–72597, 2020.
[30] M. A. Adedoyin and O. E. Falowo, "Combination of ultra-dense networks and other 5G enabling technologies: A survey," *IEEE Access*, vol. 8, pp. 22893–22932, 2020.
[31] S. Khamaiseh, E. Serra, Z. Li and D. Xu, "Detecting saturation attacks in SDN via machine learning," in *2019 4th Int. Conf. on Computing, Communications and Security (ICCCS)*, Rome, Italy, pp. 1–8, 2019.
[32] M. Perera, K. Piamrat and S. Hamma, "Network traffic classification using machine learning for software defined networks," in International Conference on Machine Learning for Networking, Paris, France, pp. 28-39, 2019. *Journées Non Thématiques GDR-RSD 2020*, 2020.
[33] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique *et al.,* "Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks," *IEEE Access*, vol. 7, pp. 34885–34899, 2019.

[34] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 7–12, 2013.
[35] E. Chemeritskiy and R. Smelansky, "On QoS management in SDN by multipath routing," in *2014 Int. Science and Technology Conf. (Modern Networking Technologies)(MoNeTeC)*, Moscow, Russia, pp. 1–6, 2014.
[36] H. Zhong, J. Sheng, Y. Xu and J. Cui, "SCPLBS: A smart cooperative platform for load balancing and security on SDN distributed controllers," *Peer-to-peer Networking Applications*, vol. 12, no. 2, pp. 440–451, 2019.
[37] C. Wang, G. Zhang, H. Xu and H. Chen, "An ACO-based link load-balancing algorithm in SDN," in *2016 7th Int. Conf. on Cloud Computing and Big Data (CCBD)*, Macau, China, pp. 214–218, 2016.
[38] Y. -L. Lan, K. Wang and Y. -H. Hsu, "Dynamic load-balanced path optimization in SDN-based data center networks," in *2016 10th Int. Symp. on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Prague, Czech Republic, pp. 1–6, 2016.
[39] W. M. AlShammari and M. J. Alenazi, "BL-hybrid: A graph-theoretic approach to improving software-defined networking-based data center network performance," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, pp. e4163, 2021.