Webology (ISSN: 1735-188X)
Volume 18, Number 5, 2021

A Hybrid Intrusion Detection Model For VANET Using SDN And Growing Hierarchical Self-Organizing Maps

K. Savitha[1] , Dr.C.Chandrasekar[2]

[1]Research Scholar, Department of Computer Science, Government Arts College, Udumalpet, Tamilnadu, India

[2]Assistant professor and Head, Department of Computer Science, Government Arts and Science College for Women, Coimbatore -45 , Tamilnadu, India

Abstract

VANET is the most promising, fast-growing and modern technology. VANETs enable Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I). The VANET's security is one of the most crucial concerns from a prediction and prevention standpoint. Machine Learning based Intrusion Detection System (IDS) can be used to sense the abnormalities, but it suffers from low detection rates, high false alarm rates, and, further, it requires massive amounts of data for prediction accuracy. This paper recommended a framework for the 5G-based VANET system that combines Software-Defined Networking (SDN) with Growing Hierarchical Self-organizing Maps (GHSOM). The suggested system will be a new mix of SDN and a Growing Hierarchical Self-organizing Maps (GHSOM)-based network solution to improve security in both dimensions, detecting and blocking assaults. The suggested method investigates the effect of GHSOM settings on the detection ratio of various types of attacks on a VANET. Experiment findings demonstrate that our technology can sense malicious network traffic efficiently, prevent and mitigate DDoS assaults, and improve VANET's security and recovery speed from assaulting traffic. Furthermore, the proposed system has a high accuracy rate. The experimental results illustrate the proposed model's usefulness and efficiency in terms of detection accuracy and other metrics investigated.

Keywords: Software-Defined Networking, VANET, Vehicle-to-Vehicle, DDoS Attacks, Accuracy

1. Introduction

The link developed between mobile vehicles is brief in VANET. Due to the in-built features of VANET like dynamic topology, large network size etc., the implementation of consistent policies

is unrealistic. For the routine VANET operation, the features of VANET also create challenging problems. The dynamic and alternative connectivity results from high mobility to minimise the connection time and routes between mobile vehicles [Mejri et al. 2014]. For monitoring and controlling traffic density and routing paths, the consistent policies that are deployed in vehicular components are used by the conventional VANET to make everything possible. Flexibility, problems in trustworthiness, persistent discarding, and distinct interfaces are the various obstructions in VANET [Chahal et al., 2017]. Different VANET applications associated with data packets routing are impacted due to these issues [Liang et al. 2015]. Besides packet collision, high mobility also causes the performance of networks and collaboration among vehicles. The solution for the enhancement in the network's results in EED, QoS, and PDR is provided and addressing the exciting problems of VANET is done by [Gawas et al. 2019]. [Halabi and Zulkernine, 2019] proposed a research work that focuses on scalability, high node density, security challenges and routing.

Recently, network management has been simplified, and novelty through network programmability is enabled by SDVN architectures that have been growing as a fruitful technology. Academia and industry have started paying their attention to SDVN. In SDVNs, the control and data planes' decoupling is enabled by the SDN technology that offers: (i) the basic networking infrastructure with an idea of VANET applications, and (ii) a networking intelligence and network state that is centralized logically. The majority of the existing challenges of VANET can be addressed by SDN's convergence with VANET that is looked at as a significant direction. Specifically, networking resources' dynamic management and effective networking services enabled by up-to-date global topology are the significant features of SDN that are used to improve the experience of the user. The cutting-edge demands of VANETs that contain low communication latency, scalability, variability, and high mobility can be met by these SDN features.

Primarily, the services in cloud computing, access networks and data centres are enhanced using the SDN paradigm. But in recent times, numerous next-generation wireless networks like Information-Centric Networking (ICN), 5G [Yousaf et al. 2017], and IoT [Bera et al. 2017] are incorporated with SDN and OpenFlow techniques. On a considerable scale testbed for live video streaming applications, SDN enables users to travel seamlessly across distinct wireless infrastructures [Kobayashi et al. 2014]. This is proved by authors in [Yap et al. 2010] after assessing the Stanford ONRC Open Roads project. The performance of SDN to make the handoff between APs and Wi-MAX stations is evident from the results, and over both the networks (i.e., Wi-MAX/Wi-Fi), it can tri cast the video streams. The computer-generated APs that intensely enhance its management is offered by cloud-MAC [Vestin et al. 2013], software-enabled architecture for enterprise WLANs.

The successful outperformance of SDN over VANET proved its excellence against others. The integration of SDN-VANET is as per the suggestion of the name. Since there is an excellent chance for SDN to be used worldwide in the future, it focuses its attention on both industry and academy. For splitting the data control, SDN offers a network architecture. This technology enhances a dynamic network where the arrangement of resources is easy with cost efficiency

because of the centralized control of this technology. The utilization of SDN controller is highly advantageous irrespective of more possibility to address VANET system. The centralized network intelligence and the separation of the primary network infrastructure from the applications are the causes of distinctions that existed between the data plane in VANET and the control plane.

The basic privacy needs should also be satisfied by the SDVNs. The SDVN architecture is required to assist in identifying the significant needs for addressing security issues in SDVNs. For example, the approved SDN controller should be accessed only through a secure northbound interface protocol. Additionally, the presence of new networking and architectural components warrants new security measures for the emerging SDVNs. SDVN are very much prone to security threats because of its layered design. This is due to the layers' high functional cooperation; threats to one layer could significantly affect the others. Hence, an efficient top-down approach is recommended as a way forward in [Akhunzada & Khan, 2017] for addressing the new security threats in SDVNs effectively.

Periodic Self-Organizing Maps (PSOM) based on an unsupervised approach to detect anomalies in interrupted time intervals was proposed in another significant study [Zhang et al. 2017]. The suggested method is employed with offline IDS evaluation, and the detection information is used with an IDS to convert HD data to 2-D data, still maintaining the linkages between clustering and topological associations. Likewise, [Huang et al. 2013] introduced an aberration detection method based on a growing ordered SOM. This method is based on two stages: (1) the data mining stage and (2) the identifying step.

Depending on the evidence discussed above, it is possible to conclude that earlier studies intended to resolve the DDoS problem in SDN and VANET. Yet, despite the fact that the SOM approach has indeed been employed in several SDN research, none of the earlier investigations, to the best of our knowledge, focused on the domain of VANET. Furthermore, most of the offered solutions relied solely on the classic SOM mechanism, which may not have overcome the restrictions of the centralized SDN design. There are several promising DDoS-prevention technologies out there, but the knowledge metrics and authentication algorithms employed by the RSU/edge devices in the vehicles have many disadvantages, which may delay the detection of an attack.

This research work presents our architecture that employs a distributed approach to solve the primary access point of link failure and provide load balancing. Next, we employ a distributed GHSOM in multiple layers to boost the intrusion detection performance while minimizing overhead.

Furthermore, the proposed technique is intended to produce a smart system capable of detecting, mitigating, and preventing potential attacks. The result section also shows that the proposed model's outcomes obtained using our distributed GHSOM approach are highly efficient.

The rest of the research paper is presented as follows: Section 2 presents the preliminaries followed by Section 3 with the proposed security architecture, Section 4 presents the results and discussion, followed by the paper's conclusion in Section 5.

2.0 Preliminaries

2.1.1 GHSOM Model

Certain limitations have to be addressed, for which the SOM is proved to be a capable device to detect structure in HD data and arrange it in accordance with a 2-D output space. The inability of capturing the integral hierarchical data structure is one of the limitations. Moreover, the map size must be fixed before when there might not be an accurate understanding of the data distribution features. But an architecture that completely adapts to the input data features is not provided by these approaches. According to the data structure, the Growing Hierarchical Self-Organizing Map (GHSOM) that appropriates its multi-layered architecture is developed to overcome the disadvantages [Dittenbach 2000].

Figure 1 gives a visual illustration of a GHSOM. In layer 1, 3x2 units are contained in the map and instead, an irregular arrangement of the key clusters in the input data is provided. A comprehensive observation of the data is given by the 6 separate maps in the second layer. The subset that has been mapped onto the subsequent unit in the top layer is nothing but the input data for one map. An adequate granular input data representation is provided by further expanding into third-layer maps by two units from one of the second-layer maps, according to the data, structure that free us from the risk of predefining the architectural structure. The whole data set's representation is layer 0, and it is essential for controlling the growth process.

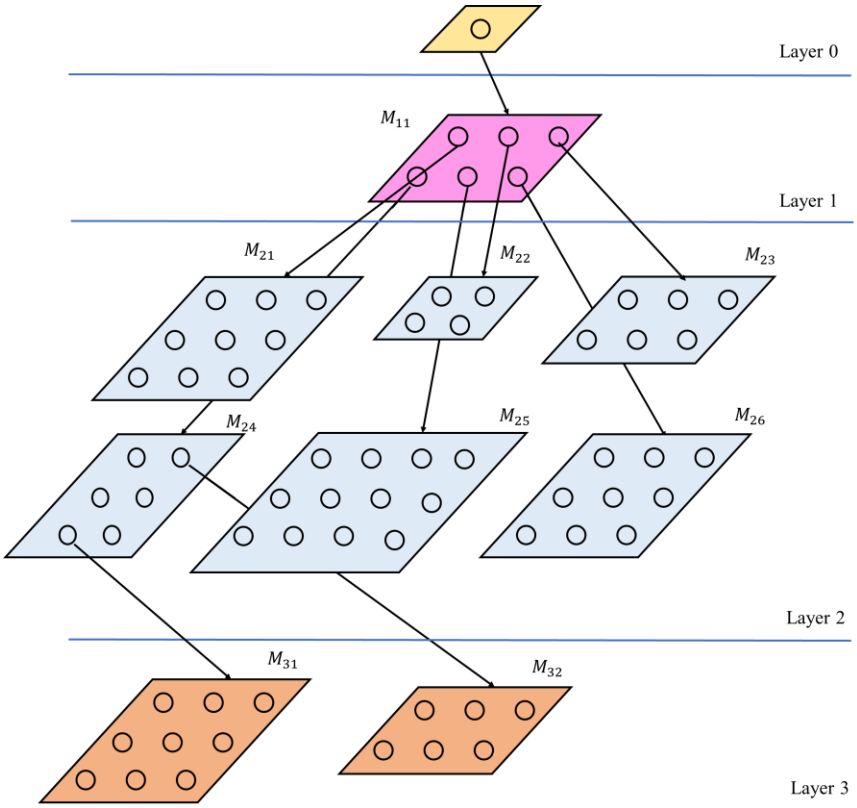

Figure 1: GHSOM Representation

2.1.2 Distributed Growing Hierarchical Self-Organizing Map (DGHSOM)

The DGHSOM framework is a multi-layered hierarchical design. Separate neuron maps are used to create the various layers of a neural network. The framework consists of a Layer_Trainer and numerous Inflators for distributed training of the GHSOM model. Inflators take the neuron map message as input and produce multiple neuron maps as part of the SOM clustering work. The Growth_Result is returned to the Layer_Trainer when an Inflator completes its operation. The result is then appended to the tree by the Layer Trainer. We use Agent models to scale-out Inflators because the training may be done independently at the same level.

A. The Agent Model of DGHSOM

An Agent method is developed for the development of a message-passing distributed system. Entities known as Agents populate an Agent-based system. With asynchronous communications, each Agent is in charge of the corresponding task. Send-n-Forget and Send-n-Wait are the two methods for sending a message to Agents. Instantaneously, both algorithms send an asynchronous message to the Agents and return. Send-n-Forget doesn't expect Agents to respond, but Send-n-Wait sends a reply in the form of a future object, which is the only thing that makes them different. So, while both methods are asynchronous, out of both, one returns something that the other does not. When messages are received, an Agent's behaviour is defined by how they handle them. Because an Agent's state is inaccessible from the rest of the system, it doesn't have to worry about synchronization issues like thread locking when changing its state.

Algorithm 1: DGHSOM	
Step 1:	Mean Vector:= Acquire Mean Vector (data Set)
Step 2:	Data Set MQE := Acquire Data Set MQE(data Set)
Step 3:	Learned Bloom Filter := Training(data Set)
Step 4:	Init Neuron : = Build Neuron(data Set MQE, mean Vec, trained Bloom Filter)
Step 5:	Init Neuron Map := Build Neuron Map (init Neuron, data Set MQE, trained Bloom Filter)
Step 6:	Init GHSOM Layer := Create GHSOM Layer (init Neuron Map)
Step 7:	GHSOM Tree Creator := Create GHSOM Tree Creator (init GHSOM Layer)
Step 8:	inflator := Create Neuron Map Actor(GHSOM Tree Creator)
Step 9:	Neuron Mapl := Create Neuron Map (init Neuron)
Step 10:	Send-n-Forget (to Info (Inflate, neuronMap1), inflator)

Algorithm 1 helps to understand that GHSOM is made up of several factors and two Agent Components: GHSOM_Tree_Creator and Inflator. The GHSOM training involves a multi-layered tree as its fundamental component, each of which comprises several neuron mappings representing the input pattern's distribution. To determine the Mean Quantization Error (MQE) for the GHSOM algorithm, we must first estimate the mean vector of the input data. The MQE value of the dataset will be used as a critical parameter for determining whether or not to continue conducting neuron

map and neuron expansion in horizontal and vertical ways, respectively. Storage space is a critical resource; in order to conserve it, a bloom filter is trained on the input dataset to reduce its size. After obtaining all of the primary data from the input dataset, we may begin constructing a neuron utilizing this data. A neuron map with a single neuron is used to create Layer 0 of the GHSOM model. Thus, the newly formed neuron is used to generate a neuron map, which is then used to generate a GHSOM layer. When Layer 0 is created, the next step is to train GHSOM_Tree_Creator, which can be used to construct Inflator. Lastly, the process of sending neuron maps to the Inflator as messages for training the GHSOM model can be initiated.

GHSOM_Tree_Creator's primary responsibility is to record training results, save them to disc, and deactivate the entire Agent system if the GHSOM tree satisfies a specified pattern. A neuron map would be sent to inflators as a message to run the SOM expansion method. After the expansion is complete, Inflators will submit the neuron map to GHSOM_Tree_Creator as a message and wait for the next mission. GHSOM_Tree_Creator is an Agent that keeps a tree-like data structure corresponding to the present training advancement and listens for specific inputs from other Agents. Growth_Result and Guard_GHSOM_Layer are the two types of messages that the builder looks for. When the message received by the builder is Growth_Result, it will mine the message for neuron maps that are trained and update the current status by tagging it to the tree. It is possible to get two values after adding a message to a tree: the current layer's expected size and the current layer's actual size.

Additionally, the function Object () {native code} would check a condition and transmit a message if the received neuron map's GHSOM layer is satisfied. The Creator agent would destroy the transmitter to free up some storage space to manage resources. Conversely, if the Creator agent receives a message to save a GHSOM layer, it serializes all neuron maps contained in the layer into a file. Again, the builder would de-serialize each bloom filter to address the memory issue. At this level, the creator would commence the metadata for the subsequent layer, specifying the number of neuron mappings present in the subsequent layer as a state for the creator to examine upon acquiring a Growth Result message. Eventually, after the entire tree structure is fully inflated, the builder checks to determine whether the Agent system can be terminated or not.

B. Inflator

An inflator is s map of the neurons that encapsulates most of the GHSOM algorithm's original functionality. As an Agent, an Inflator's sole function is to train SOM with neurons, which are also Agents. Inflator establishes a collection of Agents as neurons to maintain the weighted state by using the Inflate message as an igniter. The neuron Agents first initialize the bloom filter with a null value, and later as the process evolves, it assigns a weight vector to track the type of input patterns accepted by each neuron. Additionally, a neuron Agent is responsible for estimating the Euclidean distance amongst its weight and the chosen input sequence encapsulated in a message and constantly adjusting its weight with respect to certain parameters during the SOM training process. Upon arrival at a checkpoint during the training phase, the Inflator will request the winner of the neuron, which seems to be the nearest in Euclidean distance, to update the bloom filter for

each input pattern. In order to determine the MQE of such a neuron map, the Inflator transmits information to neuron Agents, requesting that they inform the status of each Agent containing the MQE. Once neural Agents acquire a Measure_QE input, it estimates the Euclidean distance between the random weight in the message and their own weight, and it communicates the result to Inflator. When neuron Agents get the Adjust_Weight instruction, every neuron's weight will be updated.

For the sake of memory, neurons in the old neuron map will be destroyed as the new neuron map grows. The Inflator would then use this to generate new neuron maps based on the neurons that met the tau2 requirement. The Inflator creates new Inflators based on the updated neuron mappings and delivers the metadata to each new Inflator as a message. GHSOM_Tree_Creator receives the results of the trained neuron map from the Inflator and completes the expansion.

3 Proposed Architecture

3.1 GHSOM based Distributed Multilayer Architecture

The architecture of our system is described in this section and also how the security against DDoS attacks can be enhanced by assimilating SDN over VANET networks when MGHSOM is used. For reducing the latency faced by the vehicles, the controller's location should be in close affinity to data plane nodes while installing the SDN-VANET architecture. Consequently, the latest works on the SDN-VANET aim to reduce the maximum volume of communication among the data and control planes. The RSUCs fix the typical VANET framework attributes that make the possibility to move the Internet from the control plane to the RSUC level. The connection with the controller will be reduced due to this, and communication with controllers will be enabled by using the current wireless technology rather than utilizing high-cost cellular links. Using the Open Flow protocol, the Road Side Unit Controller (RSUC) is communicated by the SDN controller, whereas the vehicles within the coverage area of the roadside controller are connected with it. Through the supporting status update of every vehicle, the roadside controller trusts the extension of the capacity of the OpenFlow standard. Integrated with the distributed controller placement, our architecture was designed to solve the load balance issue and disseminate the workload among controllers.

For the purpose of assigning tasks related to traffic processing among distributed RSU Controllers for reducing the load and processing a less traffic volume related to policy checking, the proposed architecture was specifically designed. Additionally, the traffic that comes into its ports from outside networks are controlled by each RSU in our architecture. Hence, the load on the RSU is not essential when compared to the single controller case. As a whole, the following module are contained in the proposed architecture model (Figure 2).

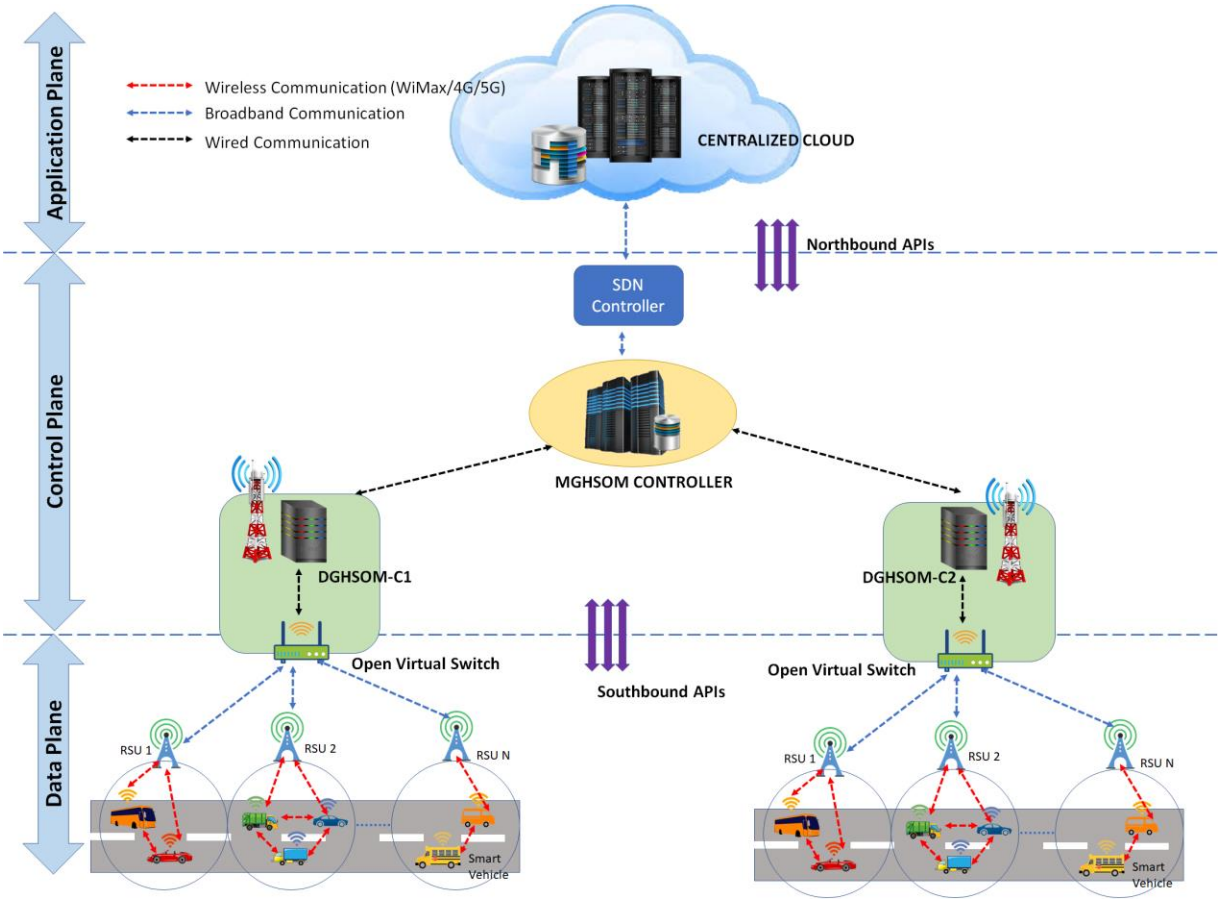

Figure 2: MGHSOM Architecture

3.2. Next Generation Node B Controller (g NBC)

In VANET, the SDN controller unit is responsible for establishing consistency, policy control, security enforcement, and traffic management. The emerged base stations of gNBC is integrated with SDN. The next generation NodeB (g NB) is positioned near the OpenFlow controller in this model. For access data, 5G UPF transfers vehicle data' traffic to the relevant gNB node encapsulated in GPRS Tunnel Protocol in the downlink direction (GTP-U). The downlink packets are decapsulated, the content is ciphered, PDCP (Packet Data Convergence Protocol) and RLC (Radio Link Control) headers are added, and the packet is then forwarded to the open flow controller. For reliable transport, gNB employs ARQ, which necessitates the storage of all downstream packets in a packet buffer until an acknowledgement is received. The packet is re-transmitted, and the appropriate acknowledgement timer is resumed if the acknowledgement does not come within a timeout period.

The distribution of workflow among RSUs and local agents of vehicles is by which our design works handling synchronization with a base controller correspondingly. In between the designed architecture, there exist various communication technologies, as shown in Figure 2. The following are the communication channels:

(i) G NBC ↔ RSUC: It is a channel for transferring rules of the control plane and established procedures.

(ii) Vehicles ↔ RSUC: A dual communication protocol controls security rules/flow management.

(iii) G NBC ↔ Vehicles: When there is no direct connection of the vehicles with RSUCs, a channel is directly connected to the g NBC.

(iv) Vehicles ↔ Vehicles is a data plane channel. The RSUs and the g NBC's installation and administration of flow rules and policies are managed by the data exchanged between vehicles. The following three interfaces are integrated by the architecture to facilitate the controller to communicate with the remaining SDN planes: (1) Northbound API, (2) Southbound API, (3) East/Westbound API.

3.3 MGHSOM Algorithm

The suggested system employs the MGHSOM algorithm to discover and mitigate DDoS attacks. This approach is used to create an accurate description of the assault and its kind, which an anomaly detector can detect. Figure 3 depicts the proposed system, which includes MGHSOM and DGHSOM modules, g NBC, RSUCs, switches and mobile nodes (vehicles). A DGHSOM component is incorporated into an RSU controller using extension modules in our design, and the MGHSOM module, which controls the entire system operation, is placed in the g NBC. The DGHSOM system is placed on the RSUC and gNBC to enable independent performance upon detection. Network disruptions during overcrowding must be determined for VANET communication media.

The suggested system's methodology is based on a distributed multilayer growing hierarchical self-organizing map built with a distributed GHSOM (DGHSOM). The system is broken down into the resulting distinct phases:

(i) DGHSOM Training: The approach begins with training the DGHSOM, which is similar to the GHSOM initialization described above. The gNBC sends the initial training dataset, which is used to train each roadside controller. This training results in input data that is structured into regions by taking into account the similarity value between each piece of data. The training of MGHSOM with a class label continues to increase the precision of our proposed system. The DGHSOM, on the other hand, already knows the similarity function, neighborhood update, and each cluster's class label. The DGHSOM algorithm has a disadvantage due to its feature, Standard Deviation (SD), that causes it to perform poorly, which is undesirable. To address this constraint, we use Kendall rank correlation instead of Euclidean distance measurement. This method has no drawback with pertain to SD and produces good outcomes.

Observations having a similar rank between the two variables will have a high Kendall correlation, while those with a dissimilar/entirely different rank between the two variables would have a lower correlation. Kendall's coefficient is defined as follows:

$$\tau = \frac{(\text{no.of concordant pairs}) - (\text{no.of discordant pairs})}{\binom{n}{2}} \qquad (5.17)$$

Where $\binom{n}{2} = \frac{n(n-1)}{2}$ is the binomial coefficient for all possible methods to select two variables from n variables.

a. **Feedback:** The following training using the DGHSOM, the results are passed into the MGHSOM in order to derive the appropriate classification features.
b. **Merging (DGHSOM):** Allowing each RSU to operate independently could result in relatively significant mapping variations for MGHSOM to provide diverse outcomes for similar input data. To avoid such map distortion, the DGHSOMs are merged into a single DGHSOM using the weighted sum approach on a regular schedule.

$$\mathrm{MGHSOM} = \sum_{i=n}^{j=1} \frac{W_j}{\sum_{i=1}^{j=n} W_i} \times (\mathrm{MGHSOM})_j \qquad (5.18)$$

c. **Updating:** The combined map is delivered to the RSUCs by the g NBC after merging the DGHSOMs. As a way to make sure that the classification happens at the switch.
d. **Classification:** Once the updation is done, the MGHSOM is used to perform classification on the data that has been given as input, and the output is sent to the controller so that it may be used for further processing.

3.3 The Architecture Proposed for SDN Controllers

The recommended SDN Controllers architecture was designed for the essentials of an enormous DDoS defense system with heterogeneous levels to get fulfilled. Rather than not retaining an absolute control by the SDN controller at the gNBC level, it represents control of traffic management and other details to the rest of the local agents and RSUs in the vehicles. Similarly, all SDN controllers share traffic control and security.

3.3.1. g NBC Level

The application layer and the control plane are integrated by controlling our scheme on the g NBC level. The OpenFlow controller module, which is responsible for running the requests in the control plane, supports the g NBC controller. The security module, global database, and g NBC agent are the classifications of the proposed architecture (Figure 3). The g NBC agent is accountable for upholding the northbound interface abstraction layer. Moreover, while the g NBC agent-based controls message among all modules, the integrated MGHSOM process is performed by the MGHSOM module. Each module uses different socket connections; as various kinds of connections are contained in our architecture. Besides, the agent manages the mitigation engine and the policy module.

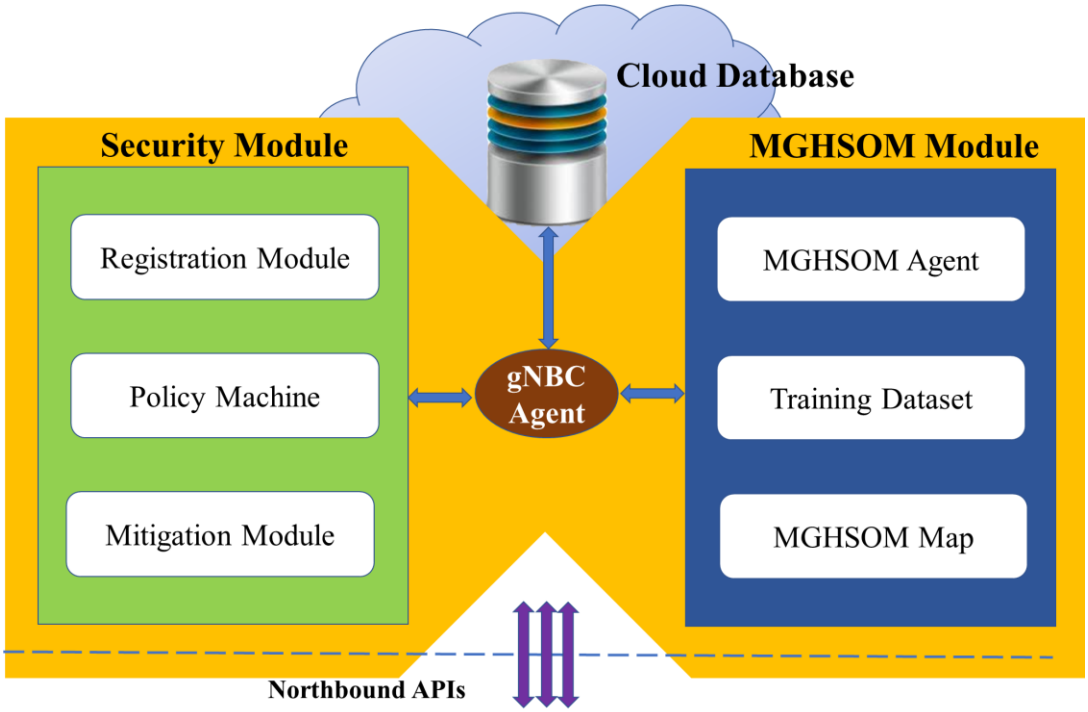

Figure 3: gNBC Agent Model

Algorithm: MGHSOM	
Step 1.	Must Run : = True;
Step 2.	Cycle : = 0;
Step 3.	While must Run Do
Step 4.	Random Weight From Data : = NextWeight (filter);
Step 5.	Info: =toInfo (Measure QE, random Weight From Data);
Step 6.	Distances : = Send-n-Wait (info, neuron Agent List);
Step 7.	Winner : = DiscoverWinner (distances);
Step 8.	Info : = to Info (Adjust Weight, winner, winner Point, Training Rate);
Step 9.	For Each neuron Agent ∈ neuronAgentList Do
Step 10.	Send-n-Forget (info, neuron Agent);
Step 11.	End For;
Step 12.	Training Rate : = Update Training Rate();
Step 13.	Neighborhood Rate : = Update Neighborhood Rate();
Step 14.	If hit Learning Condition Then
Step 15.	While (weight ∈ dataset) Do
Step 16.	If filter .holds(weight) Then
Step 17.	Winner Agent : = DiscoverWinner(weight, neuronAgentList);
Step 18.	Info : = to Info (Update Bloom Filter, weight);
Step 19.	Send-n-Forget (info, winner Agent) ;

Step 20.	End If
Step 21.	End While
Step 22.	Compute Kendall coefficient using MD Tau Sort the pairs (Xi, Yi)
Step 23.	Iterate through the pairs (X1, Y1),...,(Xn, Yn)
Step 24.	After the above iteration, measure τ using: $\tau = \frac{(\text{no. of concordant pairs}) - (\text{no. of discordant pairs})}{\binom{n}{2}}$
Step 25.	Forward merged DGHSOM parameters from RSUC to gNBC and all RSUCs
Step 26.	Classify the traffic as normal/intrusion
Step 27.	end

The management of the process of initialization, merging and updating by the MGHSOM engine is discussed above. The information related to the registered RSUCs and vehicles stored in the global database of the g NBC through its agent is retrieved by the MGHSOM module at the time of the initialization process. Besides handling the primary dataset and map, the agent also handles the process of MGHSOM performed in the g NBC. All the registered RSUs receives the primary training dataset from the MGHSOM agent after the RSUC registration process is completed at the validation and registration RSU. Eventually, the beginning stage is finished once the primary training dataset is received by the entire RSUCs. When the registered RSUCs receive the message to gather the DSOM maps, the process of MGHSOM merging commences with the MGHSOM agent. The g NBC agent quickly sends all the information regarding the DSOM map to the MSOM agent. For further merging, the MGHSOM agent piles the subsequent data after verifying all the information received from the RSUs. The above-described merging process is commenced by the MGHSOM agent after the verification process is completed. Later, the transmission of the merged map to all the registered RSUs initiates the update process, and the MGHSOM module manages the MGHSOM map.

3.3.2. The RSUC Level

Though the modules are the same in RSUC architecture (MGHSOM and security), they have distinct functions. The communication with OpenFlow switches and the RSUC architecture components are shown in Figure 4 among the modules themselves, and with the rest of the parties within the system, the RSUC channelizes the functions and handles the workflow. The beginning, classification procedures and feedback are the responsibilities of the MGHSOM module at the RSUC level. Only when the g NBC that send the primary dataset is received by the whole registered RSUCs, the training process commence.

Using the steps depicted in the proposed algorithm, the primary dataset is trained by the MGHSOM agent for creating the map. The MGHSOM save the map in the MGHSOM database after creating it. Furthermore, the merged map will be updated by the g NBC with the merged map, a novel regular request for the training, or the occurrence of any update at the time of the process

of the feature extractor. The frequent adaption of the map is assured to some of the novel arriving traffic and the performance of advances of each local MGHSOM's RSUC classification.

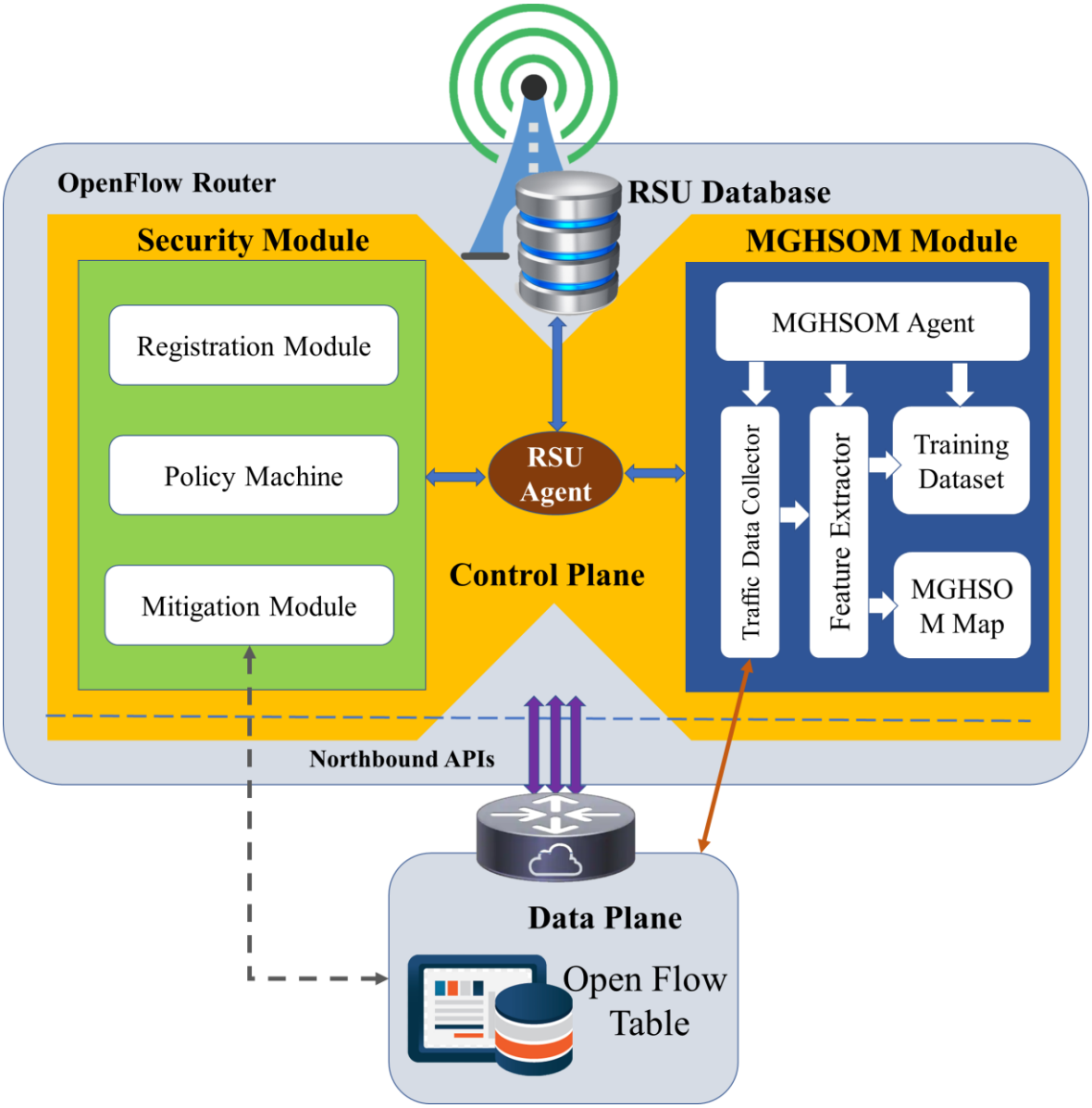

Figure 4: RSUC Architecture

3.3.3. Security Module

The requirements of VANET-based SDN networks is matched by enhancing the trustworthiness and scalability of the recommended result by the distributed controller layer. In between the multiple controller layers, the workload is distributed by the security module. The following are the workflow of the proposed security module and its elements:

(i) Registration and Authentication. The g NBC is deemed to be the central authentication component in the proposed system. Additionally, for one time, the g NBC must register and authenticate the RSUC. At the outset, every RSUC will confer a security certification

once its registration is successfully completed, and for RSUCs, they will store this certificate in the local database. But the two phases are contained in the registration of vehicles. By giving a registration request to the neighbouring RSUC, every vehicle tries to connect with the system in the first phase. To the g NBC, the received request is escalated unless the vehicle is already familiar with the second phase. After the g NBC authenticates a vehicle in the local RSU database, global g NBC database, and the car itself, a certificate is granted, and an ID will be saved. Whenever a g NBC/RSUC is connected to the system, it will assign all certified cars with a key.

(ii) Policy and Mitigation Engine: Mitigation and policy engines ascertain the defense strategy, whereas the policy engine ascertains the detection of an attack and mitigation policy. Heterogeneous mitigation plans like blocking ports, traffic redirection, and dropping packets are aided by our design. Within each element, the policy and mitigation engines perform different rules. On the g NBC that installs traffic flows and regulatory schemes, local policy decisions are made by the policy engine in the RSU. The rules are inserted directly into the flow tables by the policy engine for the purpose of the traffic collector to manage the entire incoming packets. The entire RSUs, the flow table and the vehicle policy engine will be updated by the global policy engine when the detection of attack occurs/if there is a request to modify the old policy through a REST-API command. The global policy engine feeds new rules and updates regarding every rule violation into the vehicles and RSUCs. The g NBC sends the new policy bound REST-API command whenever one or more RSUCs detect an attack, and on the basis of the network's global view, the whole network is provided with the decision made by the policy module.

The security module in the g NBC is where the mitigation module in the RSUCs works. This process commences after the MGHSOM agent sends an attack alarm to the security module. For policies updating and sending the attack definitions to g NBC locally, the policy engine module is informed by the mitigation engine module after completing verification. The attack may create a drastic number of malicious flow entries after the successful installation of the mitigation policy. Based on the flow rules furnished by the RSUC, the flow table of the vehicles is updated. Through PUSH-based REST-API, the rules are propelled to the vehicles. By introducing the spoofed packets, in which the fixed rules to a signature IP mismatch, the traceback method is initiated on the RSUC level. The mitigation module includes the following three filters based on the severity of the attack contained in the information mined from the alert message: (1) Least Reactive (LR), (2) Intermediate Reactive (IR), and (3) High Reactive (HR) filter. By dropping packets, minimizing the traffic transmitted among any two nodes and protecting against the attacks by blocking ports, these filters are used. The used memory in the switch of the vehicle agent cleared by deleting the attacker flow entries on carrying out mitigation action against the attack traffic because this malicious data flow consumes more storage space.

4. Testing and Simulation

Each module was implemented and tested independently to ensure its accuracy. The following section validates the communication layers among the agent and the RSUCs, the security module, and the g NBC. Initially, the simulation was carried out in a Ubuntu 14.04.2 based virtual computer. Mininet was employed for network simulation to simulate the RSU controller, and g NB controller POX was employed. Finally, the architecture and its attributes were simulated and tested using NS-3. The simulation was run on a road network created by SUMO. The backdrop grid is 1000×1000 m^2 in size and is divided into seven cells, and a single RSUC is employed to control each cell. The simulated region is governed by a centralized g NBC, which was installed in the middle of the network to give wireless connectivity to all RSUCs. The test consisted of 50 automobiles equipped with SDN switches capable of connecting to a single controller at a time. The traffic management module is in charge of load balancing and can transfer an adjacent vehicle's connection among RSUC to avoid overloading a single RSUC. It had two wireless connections: one for short-range (up to 250 m) and one for long-range (LTE); if the controller failed and high-priority data needed to be sent, each vehicle had both.

KDD CUP IDS dataset based on a DARPA study is employed in this simulation. Many studies on IDS use the KDD CUP 1999 dataset to test the effectiveness of an IDS. The source dataset has attributes and is categorized for different categories of attacks. There are four types of attacks that we group in this simulation:

(1) A DoS: It occurs when an attacker causes a computer to become too engaged.

(2) A Probing: This happens when an intruder tries to understand how the system works.

(3) A User to Root (U2R): It occurs when an intruder attempts to gain control of the system using a regular user account.

(4) A Remote to Local (R2L): this happens when an intruder seeks to obtain access to a computer without using a standard user account. To reduce transmission cost among the vehicle and the SDN controller, we employ 28 characteristics.

Two steps are used to simulate the proposed security mechanism: the MGHSOM training and testing phases. During the MGHSOM training phase, vehicles transmit data about traffic flow to the RSUC controller. The DGHSOM is trained by the RSUC controller and delivered to g NBC at SDN, which then trains the MGHSOM and returns it to the vehicles. During the testing phase, all vehicles use the MGHSOM classifier to categorize security attacks from the experimental dataset. The simulation contains 300 runs using a variety of random seeds. The proposed model's performance is compared to that of the algm#1: proposed by) Abdulkadhim et al. 2021) and algm#2 proposed by (Ali Hussein et al. 2017) for various scenarios as presented in the following section

4.1. Results of the Simulation

When compared to algm#1 and algm#2, we assess how well the proposed security mechanism (i.e., the MGHSOM scheme) performs. We simulate performance (i.e., precision, recall, and accuracy) in a scenario involving a variable number of cars. The following Figure 5 to Figure 7 illustrates the simulation findings. Under varying numbers of vehicle counts, we evaluate the

MGHSOM scheme's performance. As illustrated in Figure 5 to Figure 7, the MGHSOM consistently outperforms the algm#1 and algm#2 in precision, recall, and accuracy. From Figure 5, we see that the MGHSOM scheme may increase the precision of the algm#1 and algm#2 by 2.46-16.6% p and 22.76-37.4%, respectively. According to Figure 6, the MGHSOM scheme can significantly increase the recall of algm#1 and algm#2 by 2.68-12.24% and 16.38-34.8%, respectively. Additionally, in figure 7, the MGHSOM can improve the accuracy of algm#1 and algm#2, respectively, by 7.58-14.79% and 22.48-36.61%.

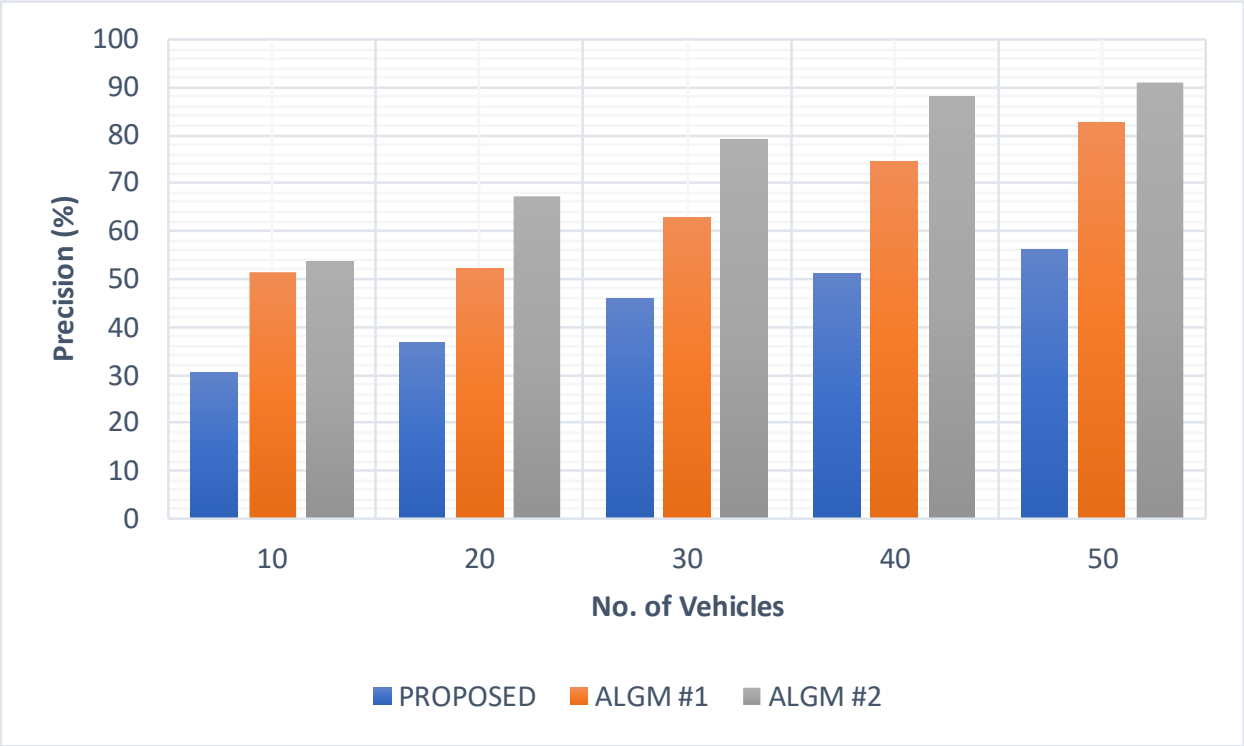

Figure 5: The impact of vehicle density (Precision)

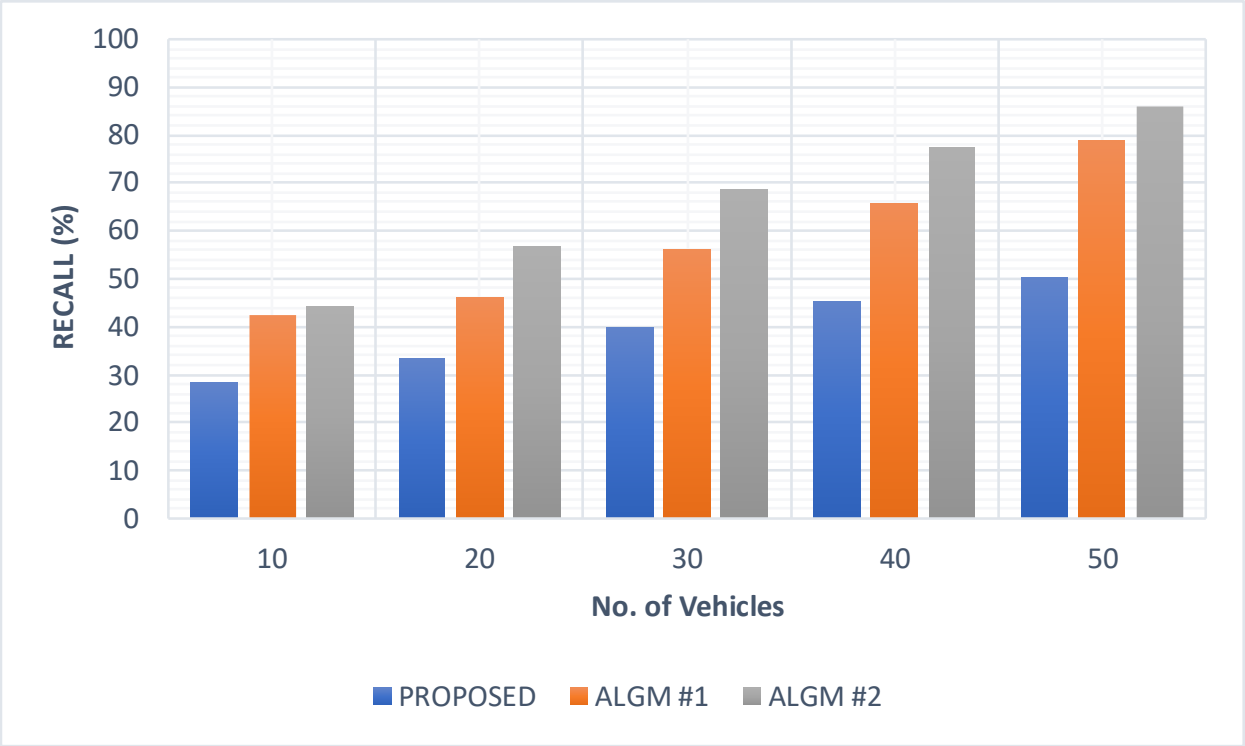

Figure 6: The impact of vehicle density (Recall)

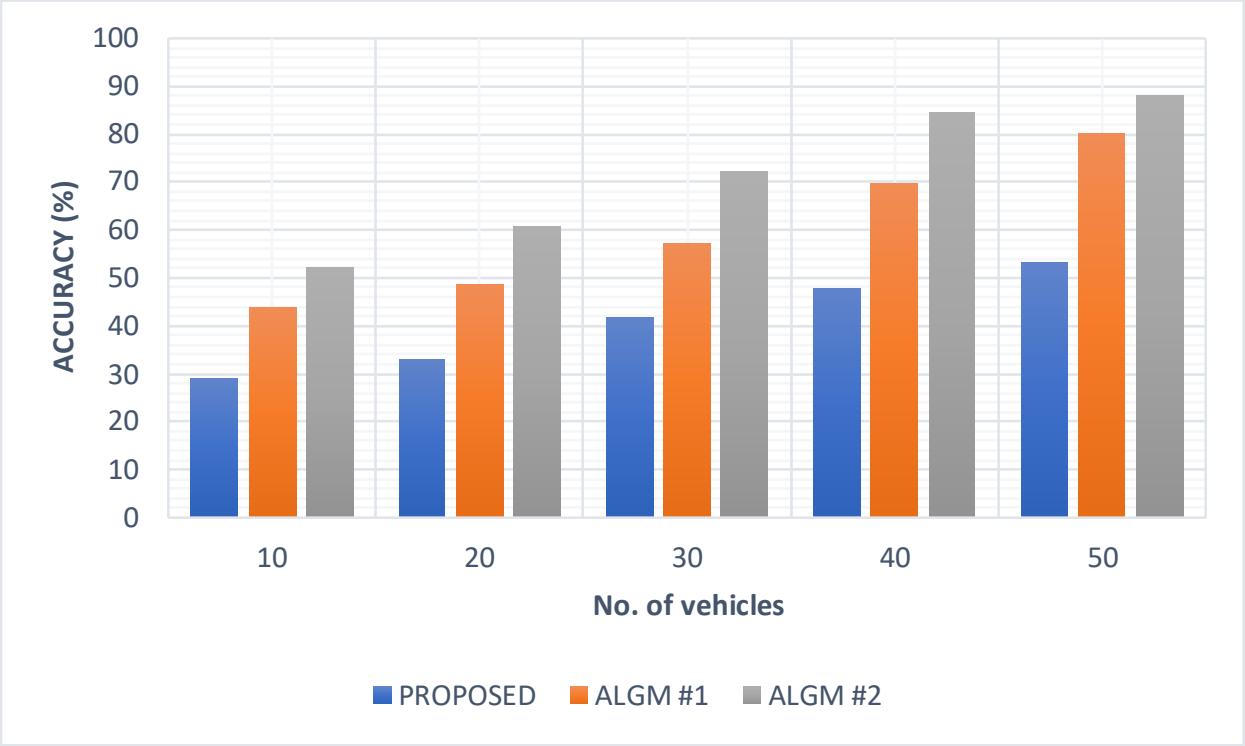

Figure 7: The impact of vehicle density (Accuracy)

The performance of the MGHSOM is also examined in a particular attack scenario. In the KDD CUP dataset, we arbitrarily shift the attack pattern duration and change the fundamental and content feature value by 10% to 50%. Figure 8 to Figure 10 depicts the simulation's outcomes. The simulation results in Figure 8 to Figure 10 show that the MGHSOM scheme consistently outperforms the other two models because the MGHSOM scheme executes SOM training in a distributed manner. For algm#1 and algm#2, Figure 8 shows that the MGHSOM scheme has an accuracy advantage of 15.39-28% and 31.94%-38% over the other two algorithms. This is also the case for the algm#1 and algm#2, which are recalled by 7.47-22.2% and 23.52-31.5%, respectively, by the MGHSOM (Figure 9). For algm#1 and algm#2, the MGHSOM can improve their accuracy by 12.26-25.9%, respectively (Figure 10).

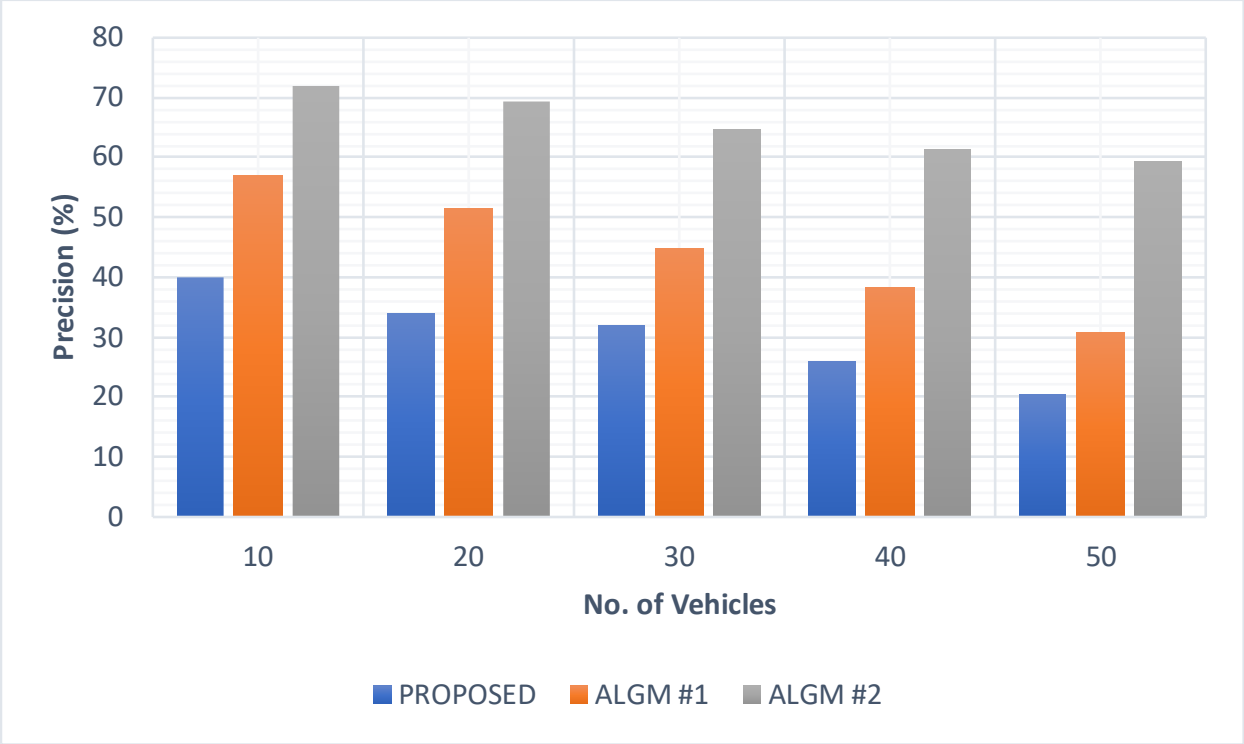

Figure 8: Effect of attack variation (Precision)

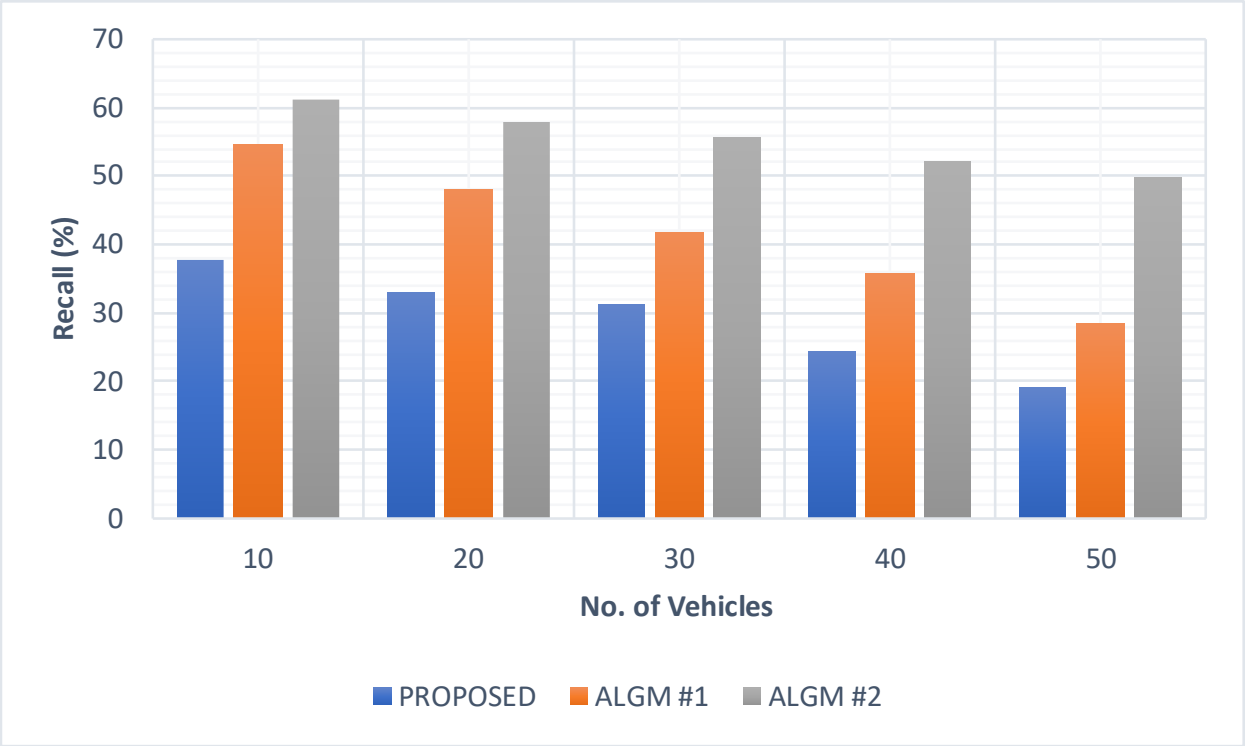

Figure 9: Effect of attack variation (Recall)

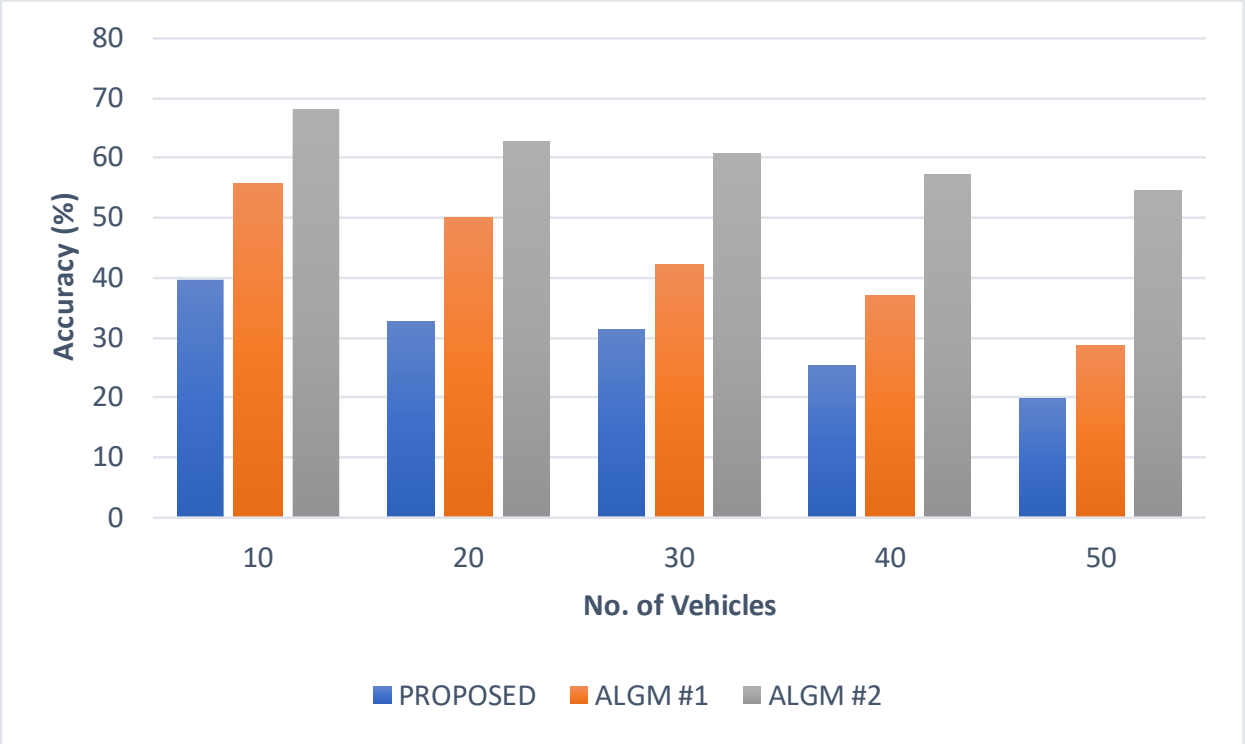

Figure 10: Effect of Attack Variation (Accuracy)

4.2. End-to-End Delay

The EED means the time consumed by Attack Alert Message for travelling in a MGHSOM model from the source to destination vehicle. There may be a delay in the on-time delivery of alert messages because of the more excellent mobility scenarios in MGHSOM. For reducing EED in VANETs, Delay-Tolerant Networks (DTNs) are chosen to be used to avoid latency in packet delivery. The better performance of the MGHSOM algorithm than algm#1 and algm#2 are detailed in Figure 11. The close consistency and a rise of 200 vehicles in every step are shown in Figure 11. The table values also depict a constant lessening in packet EED for MGHSOM with the hike in the number of vehicles. Lack of stability in the EED values recorded and the hike in the number of vehicles is apparent from the table.

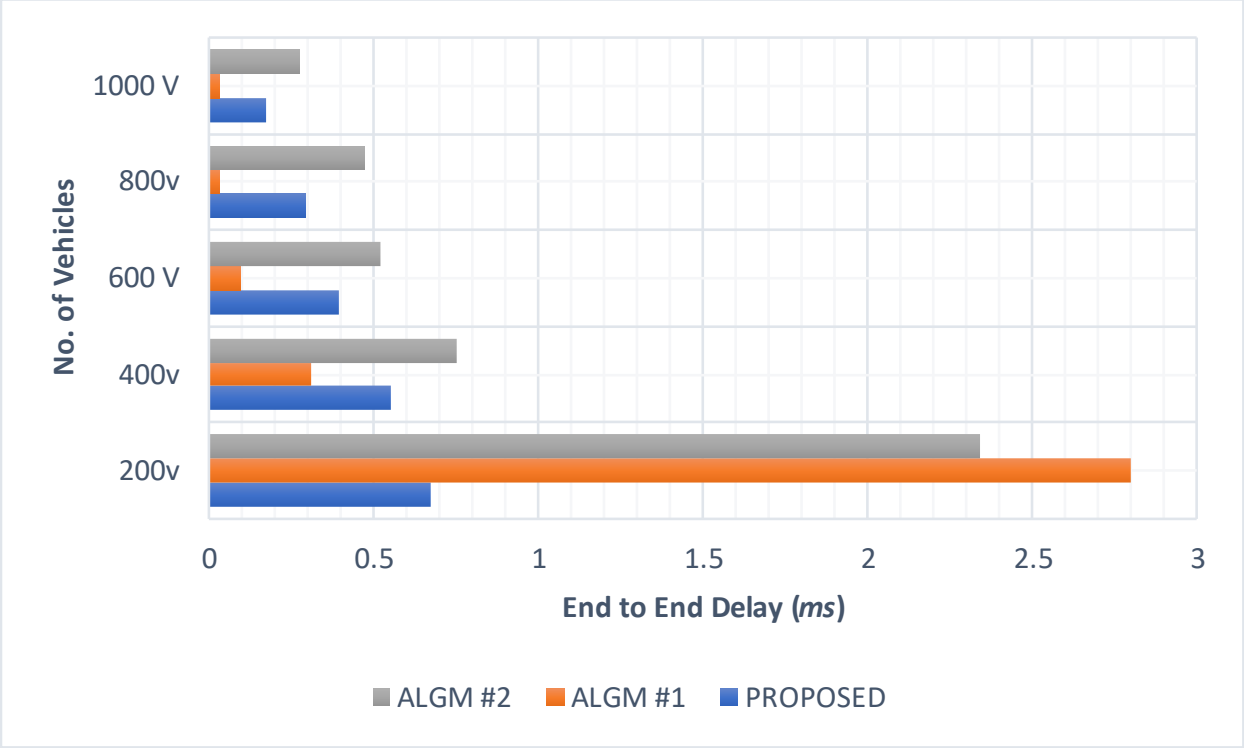

Figure 11: End-to-End Metric Performance

In the case of the MGHSOM, it is concluded that the average EED delay is roughly 0%. When compared to the MGHSOM algorithm, a prolonged delay of 57.6% and 5.2% are faced by the algm#1 and algm#2. At the same time, when compared to the algm#2 scheme, it has an EED of more than 52.4%. Hence, instead of algm#1 and algm#2, the exorbitant performance of the MGHSOM algorithm is evident from these simulation results. Because it significantly required less information regarding the network and route discovery process that restricted the network overhead and recommended as an example for dynamic and ascendable networks, MGHSOM performs better than the remaining two algorithms.

4.3. Duration of Average Link

The lifespan estimation of communication links created among the source and destination vehicles for exchanging messages is known as average link duration. For excellent performance and better data rate, the significant parameter is path choice in MGHSOMs. However, vehicle transmission range, the density of vehicle, the distance between antivehicle, vehicle transmission range are the several parameters that are depended on by the link duration in VANETs. These parameters are challenging for link duration stability. Since the verity of parameters is fundamental for link duration, the average link duration was used. Our MGHSOM's scheme is depicted as very consistent and trustworthy by Figure 12.

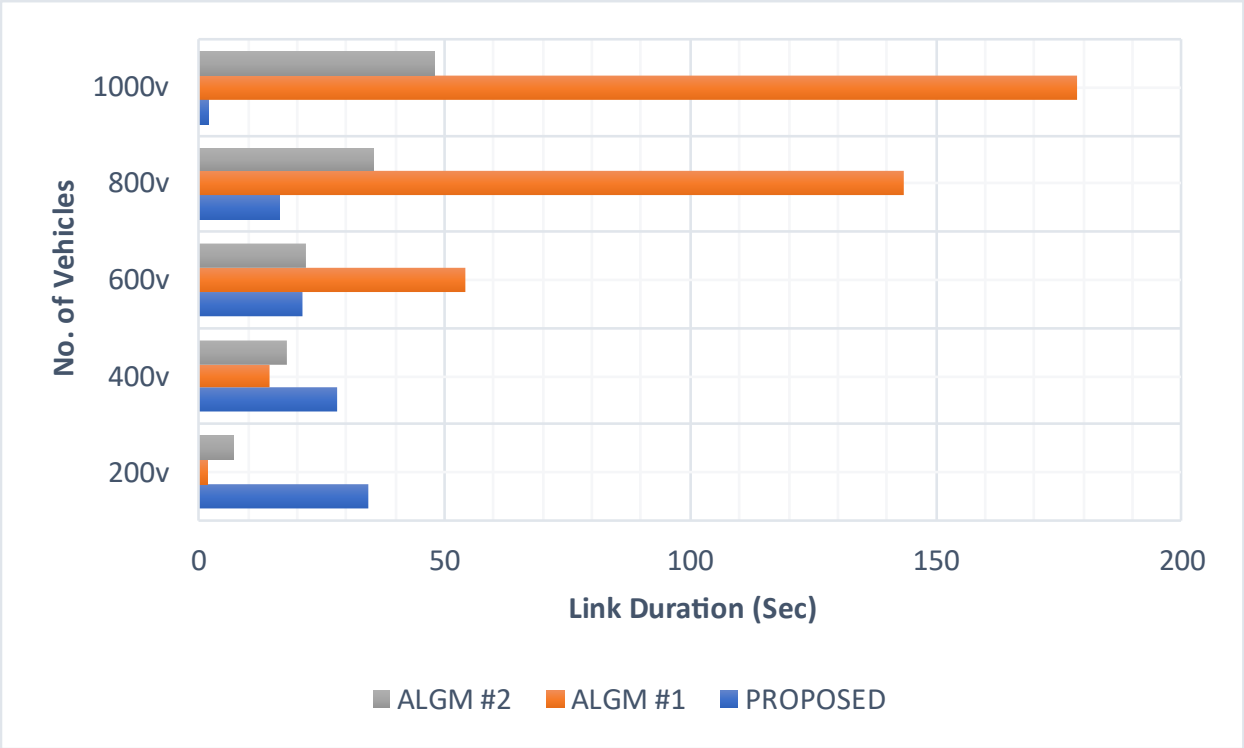

Figure 12: Duration of Average Link

An even escalation in the number of vehicles of 200 vehicles for each step exists for each step. It is concluded that when compared to algm#2 and algm#1, a highly trustworthy and consistent link duration of 29.7% and 7.8% are provided by our designed MGHSOM. But when compared with algm#1, algm#2's link duration is 21.9% more. Thus, link consistency and very little change observed in the average link values are preserved by MGHSOM with a rise in the number of vehicles. But an unexpected change in link duration with a rise in the density of vehicles and less stability that are noticed in values is experienced by the rest of the schemes algm#1 and algm#2. Therefore, better productivity in the point of view of duration present in our proposed MGHSOM is evident from the comparison results, and there is a loss of smaller packets. Among nodes for data transmission that contains timing interval of high link stability, our proposed algorithm selects and opts for more consistent and trustworthy links/routes.

4.4. Normalizing Routing Overhead

Normalized routing overhead is expressed as the ratio of transmitted routing packets divided by the total number of data packets received at the target node. MGHSOM, algm#1, and algm#2 that return an overhead are depicted in Figure 13. These schemes' overhead effects depicted in Figure 13 are shown with the rise in the density of vehicles. The density of the vehicle is adjusted at 1000 vehicles, as shown in Figure 13. With the increase in the count of vehicles, it is noticed that the proposed MGHSOM algorithm has significantly reduced in load. With the rise in the density of vehicles, there is a gradual reduction of overhead/load in the MGHSOM scheme. Whereas with the rise in the number of vehicles, there is an increase in overhead in the other two algorithms. It is concluded from Figure 13 that around 0% is recorded by overhead in MGHSOM, whereas more routing overhead viz., 27.5% and 14% are faced by algm#1 and algm#2. Therefore, the efficiency of MGHSOM is 27.5% and 14% more when compared with the other two protocols.

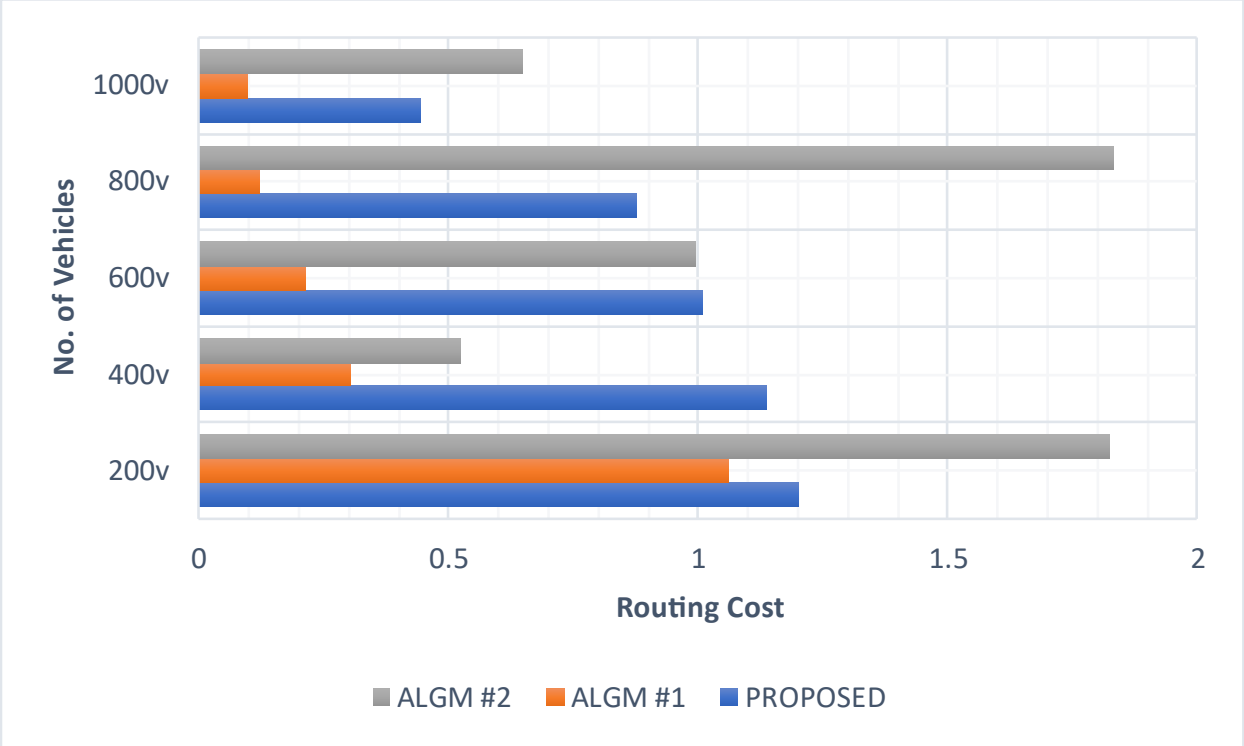

Figure 13: Normalizing Routing Overhead

Because of the fact that Route Request will be considerably reduced by our designed scheme to perceive routes and select the highly consistent and trustworthy route for data packets transmission, the specific progress is prevalent in our scheme. Fewer route failure and control messages are the results, i.e., the overhead required for detecting a route for exchanging information. A slow minimization in the values of routing overhead and the increase of 200 in vehicle density for each step, whereas an unexpected change in routing overhead is enabled by

algm#1 and algm#2 procedures and an increase of 200 in vehicle density for each step. Our scheme from this analysis outperforms the remaining two schemes.

5. Conclusion

SDVNs may represent the future of VANETs, enabling interoperability between disparate networks and effective vehicle mobility management. Additionally, the literature study revealed that SDVN continued to face security and privacy challenges related to confidentiality, authentication, and access control. Security is the primary research focus in this work when it comes to creating a secure SDVN architecture. This research proposes a novel method for securing VANETs based on incorporating multilayer GHSOM and SDN in a 5G environment without controlling planes or introducing data latency or controller congestion. The suggested solution has the potential to address critical security challenges in VANETs successfully. We conducted experiments using the KDD CUP intrusion detection dataset and employed different evaluation tests to assess the performance of the proposed system. The experimental results indicated that, as a result of the MGHSOM system's efficient adaptation to local traffic, it had a rapid system response to attacks, which improved precision and accuracy. Additionally, the suggested system's programmability enables users to incorporate a security module into the DDoS IDS approach. Additionally, the ease of management permits the fast development of appropriate mitigation and recovery methods.

References

1. A. Akhunzada, M.K. Khan, Toward secure software-defined vehicular networks: Taxonomy, requirements, and open issues, IEEE Commun. Mag. 55 (7) (2017) 110–118
2. A. Jain and D. Sharma, “Approaches to reduce the impact of DOS and DDOS attacks in VANET,” International Journal of Computer Science, vol. 4, no. 4, 2016.
3. A. Luckshetty, S. Dontal, S. Tangade, and S. S. Manvi, “A survey: comparative study of applications, attacks, security and privacy in VANETs,” in 2016 International Conference on Communication and Signal Processing (ICCSP), pp. 1594– 1598, Melmaruvathur, India, 2016, IEEE.
4. A. Suman and C. Kumar, “A behavioral study of Sybil attack on vehicular network,” in 2016 3rd International Conference on Recent Advances in Information Technology (RAIT), pp. 56–60, Dhanbad, India, 2016, IEEE.
5. Abdulkadhim, Fahad & Yi, Zhang & Tang, Chengkai & Onaizah, Ameer & Ahmed, Basheer. (2021). Design and development of a hybrid (SDN + SOM) approach for enhancing security in VANET. Applied Nanoscience. 1-12.
6. D. Miljkovic, “Brief review of self-organizing maps,” in 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2017), Opatija, Croatia, 2017

7. F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN-2014; key technology enablers for 5g networks," IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2468–2478, 2017.
8. Hussein, Ali & Elhajj, Imad & Chehab, Ali & Kayssi, Ayman. (2017). SDN VANETs in 5G: An architecture for resilient security services. 67-74.
9. J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "Cloudmac: Towards software-defined WLANs," SIGMOBILE Mob. Comput. Commun. Rev., vol. 16, no. 4, pp. 42–45, 2013
10. K. Adhikary, S. Bhushan, S. Kumar, and K. Dutta, "Hybrid algorithm to detect DDoS attacks in VANETs," Wireless Personal Communications, vol. 114, no. 4, pp. 3613–3634, 2020.
11. K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "Open roads: Empowering research in mobile networks," SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp. 125–126, 2010
12. M. Chahal, S. Harit, K.K. Mishra, A.K. Sangaiah, Z. Zheng, A survey on software-defined networking in vehicular ad hoc networks: challenges, applications and use cases, Sustain. Cities Soc. 35 (2017) 830–840.
13. M. Dittenbach, D. Merkl, and A. Rauber, "The grow ing hierarchical self-organizing map," in Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000), S. Amari, C. L. Giles, M. Gori, and V. Puri, Eds., Como, Italy, July 24-27 2000, vol. VI, pp. 15 – 19, IEEE Computer Society
14. M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. van Reijendam, P. Weissmann, and N. McKeown, "Maturing of OpenFlow and software-defined networking through deployments," Computer Networks, vol. 61, pp. 151 – 175, 2014.
15. M. N. Rajkumar, M. Nithya, and P. HemaLatha, "Overview of VANETs with its features and security attacks," International Research Journal of Engineering and Technology, vol. 3, no. 1, 2016
16. M.A. Gawas, S.S. Govekar, A novel selective cross-layer based routing scheme using ACO method for vehicular networks, J. Netw. Comput. Appl. 143 (2019) 34–46.
17. M. N. Mejri, J. Ben-Othman, M. Hamdi, Survey on VANET security challenges and possible cryptographic solutions, Veh. Commun. 1 (2014) 53–66.
18. R. Barskar and M. Chawla, "Vehicular ad hoc networks and its applications in diversified fields," International Journal of Computer Applications, vol. 123, no. 10, pp. 7–11, 2015.
19. S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1994–2008, 2017
20. T. Halabi, M. Zulkernine, Trust-based cooperative game model for secure collaboration in the internet of vehicles, ICC 2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6

21. T. M. Nam, P. H. Phong, T. D. Khoa et al., "Self-organizing map-based approaches in DDoS flooding detection using SDN," in 2018 International Conference on Information Networking (ICOIN), pp. 249–254, Chiang Mai, Thailand, 2018.
22. W. Liang, Z. Li, H. Zhang, S. Wang, R. Bie, Vehicular ad hoc networks: architectures, research issues, methodologies, challenges, and trends, Int. J. Distrib. Sens. Netw. 11 (2015) 745303.
23. S. Zhang, C. Fung, S. Huang, Z. Luan, and D. Qian, "PSOM: periodic self-organizing maps for unsupervised anomaly detection in periodic time series," in Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on, IEEE, pp. 1–6, Vilanova i la Geltru, Spain, 2017.
24. S.-Y. Huang and Y.-N. Huang, "Network dependable systems and networks (DSN)," in 2013 43rd annual IEEE/IFIP international conference on, IEEE, pp. 1-2, Budapest, 2013