Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 2989751, 23 pages
https://doi.org/10.1155/2020/2989751

Research Article

Software-Defined Multilayered Admission Control for Quality of Service Assurance in Mobile Ad-hoc Networks

Ponraj Appandairaj[1] **and Kathiravan Kannan**[2]

[1]*Department of Electronics and Communication Engineering, Easwari Engineering College, Chennai 600089, India*
[2]*Easwari Engineering College, Chennai 600089, India*

Correspondence should be addressed to Ponraj Appandairaj; ponrajaa1@gmail.com

Received 26 July 2019; Revised 19 October 2019; Accepted 9 November 2019; Published 11 January 2020

Academic Editor: Tomas J. Mateo Sanguino

Mobile Ad-hoc Network has emerged as a key technology for next-generation networks. Though its rapid growth inspires numerous applications, it is difficult to assure Quality of Service because of its immense scaling caused due to node's mobility, fading radio signals, and unreliable nature of the wireless channel. To efficiently utilize network resources and accomplish guaranteed Quality of Service, a novel Software-Defined Multilayered Admission Control model that embeds an intelligent Neurofuzzy Inference-based Admission Control service engine is proposed in this paper. Each node makes use of the Neurofuzzy Inference-based Admission Control service to learn, manage, prioritize, and admit data traffic according to user requirement. The service engine exploits fuzzy inference-based admission control process to assess node's current status using Quality of Service parameters, namely, bandwidth, queue load, and Received Signal Strength Indicator to evaluate the prediction index. The prediction index not only helps in determining the strongly connected neighbors during reliable path selection process but also solely decides whether the admission control session can be admitted or rejected. Moreover, the Neuro-Multilayered Learning process of the service engine helps to self-organize and make the complete network intelligent for instantaneous decision making. The proposed mechanism not only improves the session admission between nodes but also reduces the packet drops assuring successful session completion. Performance analysis using the simulation model proves that the proposed system shows promising gains with assured throughput and low end-to-end delay and has the potential to be applied in real-world scenarios.

1. Introduction

Mobile Ad-hoc Networks (MANET) forms a self-organized, infrastructure-less, and rapidly deployable wireless network [1]. It is constructed by a set of mobile nodes that offers unique advantage of being created and used "anywhere, anytime" but not relying on the complex infrastructure. Though MANET undergoes rapid progress inspiring numerous applications, enabling multimedia communication in mobile ad-hoc networks still remains a challenging task. Predominantly, most real-time video and audio applications makes use of stringent bandwidth typically requiring the network to support such challenging and varying traffic rates with high throughput and low end-to-end delay to guarantee Quality of Service (QoS) [2–5].

Despite the efforts made to alleviate many issues [6], there still exist a number of barriers to the deployment of real-time applications in MANET. The most prominent ones include maintaining connection state, rapid allocation of resources, avoid scarcity of resources, and effectively manage dynamic topological changes caused due to mobility and appropriate resource reservation between the source and destination. These challenges [7–9] has initiated the need to design and develop new QoS techniques [10–13] to support real-time data services in MANET. Furthermore, the new approach should be capable of differentiating various service requirements taking into consideration all other essential elements carefully with an ultimate goal to guarantee best end-to-end service delivery.

The Admission Control (AC) [14–17] is considered as the most crucial mechanism for providing QoS in MANET. Its core functionality is to estimate the node's state based on resource availability and decide which application's data flow can be admitted along which node such that the QoS as per user requirement is met without violating previously made guarantees.

To combat the aforementioned challenges and develop an enhanced technique to effectively manage network resources for assuring QoS in MANET, in this paper, novel neurofuzzy logic-based Software-Defined Multilayered Admission Control (SD_MAC) model is proposed. Aim of the proposed model is to assure guaranteed QoS support for real-time traffic. It exploits the fuzzy logic to ensure traffic adaptation characterized by dynamic changes (making it adequate to the uncertainty and the information incompleteness) in MANET environment. The model also incorporates the benefits of Neural Networks (NN) to support the multimedia service in ad-hoc networks. The self-adaptive data driven mechanism of the neural network allows fast learning of different operations performed by the core functional blocks of the system thereby permitting a best traffic class selection and delivery according to flow requirements. After learning the data presented to them it offers the capability to generalize, i.e., it often predicts an event occurrence even if the input data is noisy and their underlying relationships are unknown or difficult to describe. The most important feature of the NN is that it can approximate multivariate functions with superior prediction accuracy and low computational complexity. It can primarily be used when limited data sets are available and the relationships between them are vague. Overall NN exploits the capability to learn and adapt to the system behaviour more quickly by integrating the operations of various components (both in MAC and network layer) to make the network self-organized and artificially intelligent.

The proposed SD_MAC system makes use of the Neurofuzzy Inference-based Admission Control (NFAC) service engine whose core functionality is to

(i) Evaluate the resource availability by using QoS parameters such as bandwidth, queue load, and RSSI in each node to generate the *node status*. The Fuzzy Inference-based Admission Control (FIAC) module in the NFAC engine is triggered automatically in each node to generate the *node status*.

(ii) Derive the *prediction index* which solely decides whether the admission control session can be admitted or rejected via the node for the requested QoS.

(iii) Self-learn the behavioural functionalities, and train and test the network to make it self-organized and artificially intelligent for effective decision making using Neuro-Multilayered Learning (NML) process.

Our proposed methodology was modelled using prototype developed using MATLAB. The performance of the proposed model is evaluated by comparing it with other existing approaches such as SWAN, StAC, DACME, and FuzzyQoS. The results showed that the service exhibited by NFAC engine tends to automate the working procedure of admission control to achieve good throughput by selecting strongly connected neighbors and reliable path for longer live transmission.

The rest of the document is organized as follows: Section 2 presents related work, Section 3 illustrates our proposed SD_MAC system, Section 4 depicts the simulation results and analysis, and finally Section 5 concludes the paper along with a discussion on future directions.

2. Related Works

This section describes briefly some of the main admission control model proposed for wireless MANET.

A stateless service differentiation AC model (SWAN) [18] performs real-time traffic control using sender-based admission control by distinguishing real-time and best-effort traffic. The source probes the path to the destination when it wants to send a real-time traffic to destination. During the probe, it identifies the available bandwidth and relies on feedback received from the MAC layer by using the rate and source-based admission control approach. Each node uses the rate control algorithm to control the best-effort traffic. The bandwidth used for best-effort traffic is restricted so that real-time applications can exploit the required bandwidth; the bandwidth not used by real-time applications can be exploited by the best-effort traffic. The probe request in SWAN causes a lot of overhead and packet loss.

In Staggered Admission Control protocol (StAC) [19], the source sends the session request packet. The session is rejected if the network layer does not support the flow, else it broadcasts the request. Intermediate nodes add information to request after verifying their local resources. The destination receives the request and sends the response. Source caches all the route information and then sends an admission request packet to two hop neighbors which sends an admission denied back if it cannot support new information flow. Based on the observation, the source transmits data at a very low rate, and gradually increases the flow rate up to the required level within the specified time. During the gradual increasing time if it is affecting the earlier admitted sessions then a session may be rejected. Presence of interference causes link failures resulting in frequent route discoveries. This deteriorates the performance in StAC by incurring high routing overheads and end-to-end delay.

A distributed admission control for MANET environments (DACME) [20] approach periodically assesses end-to-end conditions on the path. Source node performs path probing to obtain various QoS measurements to assure transmission of data traffic under good conditions. The destination responds to probes giving the source feedback about path conditions. Though this method tends to perform well in small to medium network taking advantage of the IEEE 802.11e standard [21], it suffers from fairness issue under high network load.

The FuzzyQoS [22] approach aims to improve the traffic regulation rate and congestion control of multimedia applications. It monitors the variation rate in queue to provide a measure of the queue state and uses fuzzy thresholds to adapt the traffic transmission rate under dynamic conditions. It makes the source node more responsive to any sudden changes in network traffic volume by using explicit rate congestion notification. Though this approach achieves stable end-to-end delay under different network conditions, it does not deal well with route failures. It reduces the data rate of affected session while searching for new routes implying its support for real-time applications only with elastic throughput requirements.

To increase the user's perceived QoS, we apply an NFAC service mechanism in the SD_MAC model. Detailed illustration of the proposed approach is elaborated in Section 3.

3. System Description

3.1. Overview of the SD_MAC Model. The proposed Software-Defined Multilayered Admission Control (SD_MAC) model exploits a new cross-layer QoS solution for mobile adhoc networks. The overall architecture of the proposed SD_MAC model comprises of various functional blocks such as *routing service*, *admission controller*, *classifier*, *scheduler*, *and traffic regulator*.

Figure 1 illustrates the schematic representation of the SD_MAC model with its core functional blocks.

The *Admission Controller* module is considered as the most important functional block of the SD_MAC model. Its core functionality is to send a dedicated probe request from the source node towards the destination node to detect most reliable route by estimating intermediate node's resources availability. Based on the information, the admission control module at the source decides whether to admit or reject the data service request. In the proposed model, each node initiates "admission control" process to evaluate its current resource availability, which is indicated using the "service offered prediction index, $\text{SOfd}_{\text{prediction\_index}}$" value. Every node stores the $\text{SOfd}_{\text{prediction\_index}}$ value in its Local Aware Table (LAT). Next, using the "*Routing Service*" module, each node broadcasts a beacon message (consisting of the signal qualities, type of service offered ($\text{SOfd}_{\text{prediction\_index}}$), etc.) over the network. The neighbor nodes receive the beacon message and store the sender's id, service offered ($\text{SOfd}_{\text{prediction\_index}}$) by the sender, etc., in its LAT. Now, let us assume that a source node has data to be transmitted to a destination node in the network. Let us consider that a reliable route from the source to the destination does not exist, and then during such occasions, the proposed SD_MAC model performs the following steps:

(i) The source initiates admission control mechanism within itself to generate service-requested prediction index ($\text{SReq}_{\text{prediction\_index}}$) value

(ii) The source then performs the "*path selection*" process to reach the destination. As part of the "*path selection*" process, the source and the neighbor nodes looks-up its LAT to find strongly connected neighbors, SCN (nodes that have $\text{SOfd}_{\text{prediction\_index}} > \text{SReq}_{\text{prediction\_index}}$) that has the capability to process the data traffic as per user's QoS requirement

(iii) The source then sends a dedicated probe request (DPReq) message only to the selected SCNs, which in turn repeats the process until destination is reached

Unlike other admission control techniques, the source node in the proposed model does not execute admission control process every time to select the strong neighbors, rather, it simply looks up its LAT to identify the SCNs (each node maintains the list of neighbor nodes with its "*prediction index*" value in its LAT). This mechanism of validating node's resource availability using "*prediction index*" prevents the node from executing the "*admission control*" process every time a resource verification request arrives from other nodes. In addition, as every node maintains the list of neighbor nodes prediction index in its LAT, the neighbour, which has the capability to process the request, is identified instantaneously during path discovery. Moreover, to optimize traffic regulation and control congestion over the network for both real-time (audio/video) and nonreal-time services, the proposed model embeds a Neurofuzzy Inference-based Admission Control (NFAC) service engine to exploit fuzzy inference approach integrated with neural artificial intelligence. The aim is to guarantee best end-to-end service delivery for end users. Though, NFAC's core objective is to learn, manage, prioritize, and admit data traffic according to user's QoS requirement, it also keeps the network trained and tested on various behavioural changes making it self-organized and artificially intelligent for effective decision making over a period of time. Significantly, this not only improves the service flow acceptance between nodes but also reduces the dropping of service among them. The other functional blocks of the SD_MAC model such as the *classifier*, *traffic regulator*, and the *scheduler* are specifically used to distinguish, regulate, and schedule data traffic appropriately. Whenever a data service request is admitted, the *classifier* provides service differentiation by classifying and marking the packets into 4 different classes. It marks the packets either as High Profiled (HP) or Medium Profiled (MP) or Low Profiled (LP). The marked packets are then queued in their respective HP or MP or LP queues and wait for the transmission. The main objective of the classifier is to distinguish the traffic and determine the ones that should pass or bypass the *Traffic Regulator*. The *Traffic Regulator* module regulates MP and LP traffic by delaying MP and LP packet processing and allows more bandwidth for HP packet processing. When a new real-time flow is allowed by admission control block, the packets marked as HP by passes the traffic regulator and are queued in HP queues for processing. The *Scheduler* module employs a new scheduling discipline referred to as the Strict and Probabilistic Priority Queuing (SPPQ) which determines the probability with which the queue is served by assigning a parameter to each priority queue. It offers variety of service differentiation by providing service segregation among various groups of traffic classes.

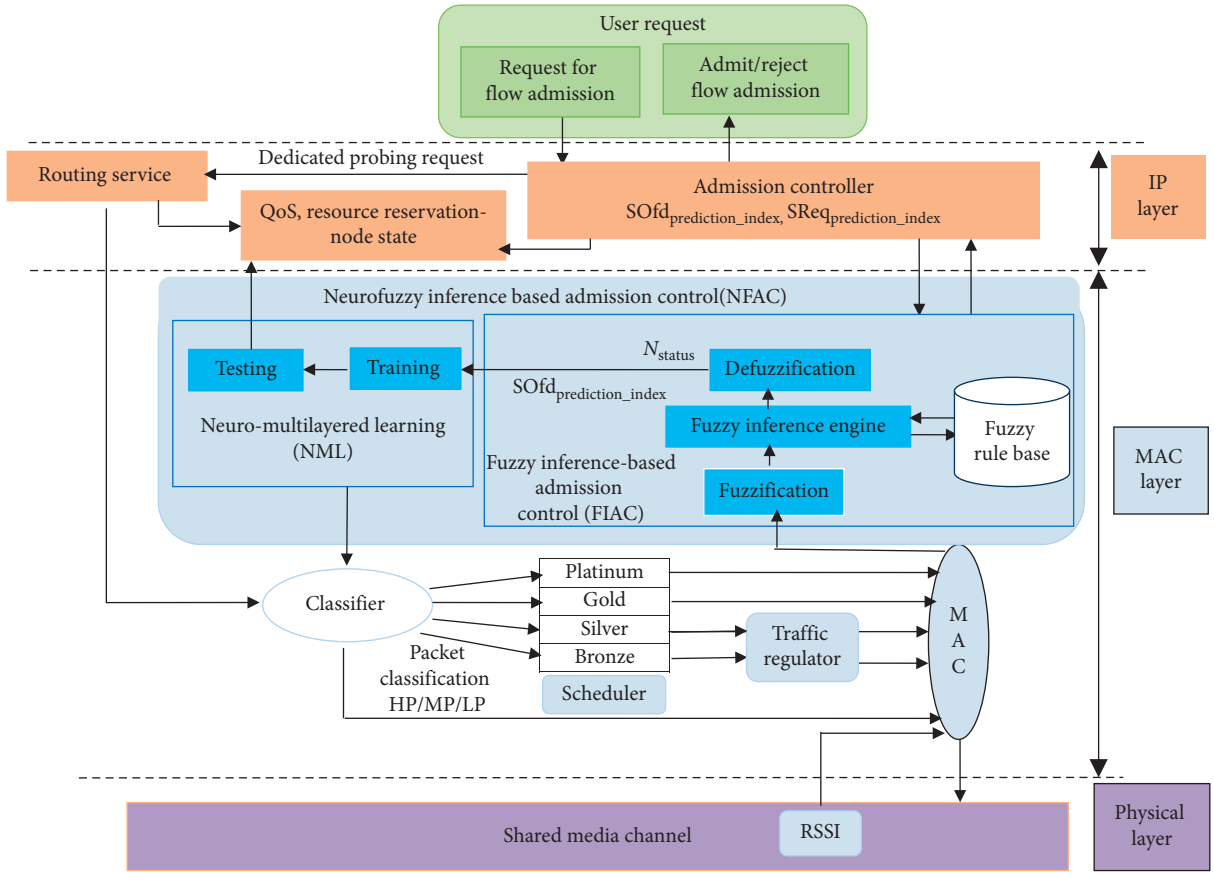

Figure 1: Model of SD_MAC

The functional blocks or components that belong to the SD_MAC system are as follows:

Routing Service: it discovers legitimate path between the source and the destination. It performs

(i) Path Selection
(ii) Resource Reservation

Admission controller: the Admission controller in the SD_MAC system incorporates a Neurofuzzy Inference-based Admission Control (NFAC) service engine to

(i) Evaluate and maintain the *node status* information for each node
(ii) Derive the certainity factor or prediction index using the node status information to solely decide whether the admission control session can be admitted or rejected

Primary functionality of NFAC is to perform the following activities:

(i) Fuzzy Inference-based Admission Control Process
(ii) Neuro-Multilayered Learning Process

Classifier: it classifies and stamps packets based on their priority (as platinum, gold, silver, and bronze) when a new request is admitted by the admission controller

Scheduler: it uses the scheduling algorithm in order to push the marked packets in their respective high- , medium- , and low-profiled queues for transmission

Traffic Regulator: it regulates data traffic in MP and LP queues if there is delay encountered in processing HP packets and vice versa

The summary of notations used in the SD_MAC system is referred in Table 1.

3.2. Detailed Specification of Software-Defined Multilayered Admission Control System. Our work primarily focuses on Routing Service and Admission Control component.

3.2.1. Routing Service. To realize service access anytime and anywhere, the path between the source and destination should be strongly connected. SD_MAC follows the Ad-hoc on Demand Distance Vector Routing (AODV) protocol for path establishment. It uses Signal Strength (SS) metric to make sure that the data traffic service is performed only on reliable links [23] rather than on weak, diminishing links (ie., selecting "reliable" links ensures that the path between the source and the destination is strongly connected). The "routing service" component performs key operation such as path selection and resource reservation.

Table 1: Summary of notations used in the SD_MAC system.

Notation	Description
S_n	Source node
D_n	Destination node
N_{id}	Node identity
NH_{id}	Neighbor node's identity
N_{SS}	Node's signal strength
N_{SNR}	Node's signal to noise ratio
N_{TS}	Node's time stamp
NHAv_{SS}	Neighbor node's average signal strength
NHAv_{SNR}	Neighbor node's signal-to-noise ratio
NH_{TS}	Neighbor node's time stamp
DPRep	Dedicated probe replay
$\text{LAT}_{\text{fields}}$	Local aware table fields
HP	High-profiled user
MP	Medium-profiled user
LP	Low-profiled user
$\text{Service}_{\text{path}}$	Service path
$\text{SReq}_{\text{prediction\_index}}$	Service-requested prediction index
$\text{SOfd}_{\text{prediction\_index}}$	Service-offered prediction index

3.2.2. Path Selection. The path selection is performed before the source S_n transmits data to the destination D_n. The process involves the following steps:

Step 1: at regular intervals, each node broadcasts the "P_beacon" UDP packet to its immediate neighbors announcing the link qualities and services provided. The "P_beacon" packet contains the following information:

$$P\text{\_beacon} = \left[N_{\text{id}}, N_{\text{RSSI}}, N_{\text{SNR}}, N_{\text{TS}}, \text{SOfd}_{\text{prediction\_index}}\right], \quad (1)$$

where N_{id} indicates the node id, N_{RSSI} indicates the node's Received Signal Strength Indicator (RSSI), N_{SNR} indicates the node's Signal-to-Noise Ratio (SNR), N_{TS} indicates the node's time stamp, and $\text{SOfd}_{\text{prediction\_index}}$ indicates the node's service-offered prediction index value.

Step 2: the neighbor nodes receive the P_beacon packet, extracts the sender node's information, and adds the entry in its Local Aware Table (LAT). Each node maintains a LAT which contains the list of its one-hop neighbor information. Entry in LAT has the following fields:

$$\text{LAT}_{\text{fields}} = \left[\text{NH}_{\text{id}}, \text{NHAv}_{\text{RSSI}}, \text{NHAv}_{\text{SNR}}, \text{NH}_{\text{TS}}, \text{SOfd}_{\text{prediction\_index}}\right]. \quad (2)$$

Each node generates and maintains a service-offered ($\text{SOfd}_{\text{prediction\_index}}$) prediction index which indicates the type of service offered by neighbor nodes. The $\text{SOfd}_{\text{prediction\_index}}$ derived using NFAC process is elaborated in detail in Section 3.3.

Step 3: in a similar way, all the nodes in the network share among each other their signal qualities and services offered using the "P_beacon" packet which in turn is added and maintained in their respective LAT. Now, using the information in LAT, each node has the capability to select its reliable neighbor nodes. This ensures each node to selectively send service queries only to strongly connected neighbors (SCNs), which may in turn forward service queries to their SCNs.

Step 4: when a source node has data service (DS) to be transmitted to a destination and if it does not know a "reliable route" to reach the destination, it first looks up into its LAT to verify if there are SCNs, which offers the required service. Typically, the *data service* consists of the following information: (i) the type of data traffic, TR_{Type} and (ii) measure of data traffic, M_{Data}.

Using the information of DS such as TR_{Type} and M_{Data} and the *node status* (N_{status}), the node derives a $\text{SReq}_{\text{prediction\_index}}$ value, where N_{status} indicates current node's status which is evaluated using FIAC's process of NFAC and $\text{SReq}_{\text{prediction\_index}}$ indicates the type of data service to be processed by the source which is evaluated using NML's process of NFAC. Details regarding N_{status} and $\text{SReq}_{\text{prediction\_index}}$ are elaborated in detail in Section 3.3.

Step 5: the source looks up its LAT to verify if there are neighbors, which satisfies the criteria: $\text{SOfd}_{\text{prediction\_index}} \geq \text{SReq}_{\text{prediction\_index}}$. If there are nodes that satisfy the criteria, then those nodes are considered as strongly connected neighbors (SCN). The source then selectively sends a dedicated probing request packet DPReq (S_{addr}, D_{addr}, Seq_{id}, $\text{Service}_{\text{path}}$, and $\text{SReq}_{\text{prediction\_index}}$) only to those SCNs. The fields S_{addr}, D_{addr}, and Seq_{id} indicate the source address, destination address, and sequence identifier. The $\text{Service}_{\text{path}}$ refers to the service path with first address filled with its own address and the address of SCNs along with its signal strength. During the LAT look-up process, the following is performed:

(i) If only one SCN satisfies the criteria, then the DPReq packet is sent only to the SCN found.

(ii) If multiple SCNs satisfy the criteria, then the DPReq packet is send to "k" SCNs whose $\text{SOfd}_{\text{prediction\_index}}$ value is high compared to others. Let "k" indicate the optimal number of paths selected for data transmission. Our scenario considers "$k = 3$," i.e., maximum of three reliable paths are selected during the "path selection" mechanism, such that when a path discovered fails during data transmission, the next path can be considered for transmission. In cases, where all three routes selected fails during data transmission, the process of "path selection" is initiated once again. An optimal limit of three paths has been chosen in our work as too many paths can increase the complexity during maintenance.

(iii) If no SCN satisfies the criteria, then the node waits for a time period and then performs the LAT look up once again. The time slot is not a fixed value and keeps varying as it depends on the node's LAT

update, i.e., the LAT in each node is updated whenever a "*P*_beacon" packet from the neighbor is received. Due to dynamic change in topology, there is possibility that each node receives "*P*_beacon" packets from new or old neighbors and records the new or updated information in its LAT. As soon as the LAT records a new entry or updates the existing entries, the node embedded with SD-MAC functionality automatically reinitiates a LAT look-up process. Moreover, the influence that this behaviour across the network is low, as in most cases, the occurrence of this behaviour in MANET is found to be minimal or negligible.

Step 6: Upon receiving DPReq packet, the SCN checks the query_handler (QH) table. The QH table maintains the details related to query handled by each node. If Seq_{id} in DPReq packet exists in QH table, then the request is considered duplicate and discarded otherwise it is forwarded to its SCNs. It also sends a Dedicated Replay Acknowledgement packet (DRAck) to the source indicating the status on path selection and for decision making by the source. The process continues until DPReq packet reaches the destination node. Algorithm 1 summarizes the steps involved in the path selection process in the SD_MAC model.

3.2.3. Resource Reservation. On receiving the first DPReq packet, the destination initiates the resource reservation process. It generates a Dedicated Probe Replay (DPRep) message and sends it along the reverse path. The intermediate SCNs which receives the DPRep packet revalidates its resource availability by checking its current $\text{SOfd}_{\text{prediction\_index}}$ ($\text{SOfd}_{\text{prediction\_index}} \geq \text{SReq}_{\text{prediction\_index}}$), and if satisfied, it updates its node state (N_s) as *reserved*. Note, each node maintains a "Node State-N_s" which helps in detecting and recovering QoS violations. At any particular instant of time, the N_s exists in any one of the following states:

(i) *checked*: if path is selected during path discovery

(ii) *reserved*: if node reserves resource during the resource reservation process

(iii) *allocated*: if node receives data packets and processes

(iv) *liberated*: if resources are released by the node

Resources reserved by the node can be liberated if data packets are not received by the node within a short allocated period (the time slot is set specific to the application within which if the neighbor node does not receive the data packets from the source or from its neighbor, it will release the reserved resource for other purposes. This ensures that the resource reserved for data transmission in the neighbor nodes is utilized optimally rather than being locked for performing a particular task). After sending DPReq along multiple SCNs, the source may receive multiple DPRep response from different service paths. It is now the responsibility of the source to select the best service path to deliver the data traffic. Our SD_MAC model selects the path with least service path discovery time by always choosing the first DPRep response path because this is the path that has experienced the least delay and is probably less congested and shorter than the later arriving DPRep response path. By using a dedicated control session, the source selects a reliable path (a path that is established for an acceptable period for transmission) with intermediate SCNs that have the capability to serve source node's QoS requirement. Now, the source admits data traffic along the selected path to destination.

Figure 2 represents the sequence diagram depicting the steps performed in a routing service component.

3.3. Neurofuzzy Inference-Based Admission Control (NFAC) Mechanism. The SD_MAC model makes use of the fuzzy neural approach based admission control to exhibit better efficiency which leads to higher user's satisfaction. The Neurofuzzy Inference-based Admission Control (NFAC) engine combines the linguistic control capabilities of the fuzzy logic controller and the learning capabilities of the neural networks to develop an intelligent system to handle incoming traffic keeping the resource utilization at an optimal level.

The Neurofuzzy Inference-based Admission Control (NFAC) engine acts as a key component of the SD_MAC model. The proposed model embeds the NFAC service engine, an automated process triggered in each node to

(i) find the scheduling mechanism and the type of services being used to estimate the current data traffic serviced and the actual bandwidth available for accommodating other data services

(ii) Find its current availability by observing and predicting the network conditions

(iii) Identify the amount of resources utilized for high- , medium- , and low-priority packets. Its core objective is to generate the "*node status*, N_{status}" and thereby derive the "*prediction index*" which determines whether the node has the capability to accommodate the type of services requested by other nodes in the network.

In SD_MAC, the Admission Control module maintains the *prediction index* value which helps each node to dynamically regulate real-time data transmission and accurately allocate resources in the face of network dynamics such as mobility and traffic overloads. Unlike other AC approaches, which executes the AC process to estimate the node's current status, in SD_MAC the decision regarding the node's current resource availability is made instantaneously by merely verifying the "prediction index." NFAC's functionality is categorized as follows:

(i) Fuzzy Inference-based Admission Control Process

(ii) Neuro-Multilayered Learning Process

3.4. Fuzzy Inference-Based Admission Control (FIAC) Process. The NFAC service engine is empowered with the Fuzzy Inference-based Admission Control (FIAC) process which

```
// Node i broadcasts P_beacon packet to immediate neighbors
N_i ⟶ P_beacon (N_Id, N_RSSI, N_SNR, N_TS, SOfd_prediction_index);
// Neighbor Node j receives P_beacon packet of N_i node i
Nj ⟵ P_beacon (N_Id, N_RSSI, N_SNR, N_TS, SOfd_prediction_index);
//extract the beacon of Node i and store in LAT of Node j
LAT_j ⟵ extract_data();
N_j ⟶ send P_beacon (N_Id, N_RSSI, N_SNR, N_TS, SOfd_prediction_index);
N_i ⟵ P_beacon (N_Id, N_RSSI, N_SNR, N_TS, SOfd_prediction_index);
LAT_i ⟵ extract_data();
/ * entry in LAT contains the following field. ?/
LATfields = [NH_id, NHAv_RSSI, NHAv_SNR, NH_TS, SOfd_prediction_index]
// if source node has data and does not know path to destination
if (S_n (data) = = "T" && path_to_destination( ) = = "F") then
SCN_i = S_n_lookup (LAT_sn);
end if;
if (is exist(SCN_i)) then // strongly connected neighbor node exists
   // send DPReq packet to SCN found
   S_n ⟶ send DPReq (S_addr, D_addr, Seq_id, Service_path, SReq_prediction_index)
   // SCN receivesDPReq packet
   SCN_i ⟵ DPReq (S_addr, D_addr, Seq_id, Service_path, SReq_prediction_index)
   Check_QHtable(); //SCN checks QH table
   if (Seq_id = = exists()) then
        Discard DPReq();
   else
      if (SOfd_prediction_index ≥ SReq_prediction_index) then
         append (Servicepath);
         extract_signal_data();
         store_into_LAT();
         forward DPReq() packet to SCN;
         send DRAck() to source //send Ack to source
      end if;
   end if;
end if;
```

Algorithm 1: Path selection process in the SD_MAC model.

comprises of fuzzy logic (FIL) [15] that offers better adaptability under varying network conditions by tuning rules without intervention of operators. Aim of FIAC is to formulate a control decision by considering multiple QoS parameters. The Structure of FIAC used in NFAC is shown in Figure 3.

Our scenario considers QoS parameters such as

(i) Type of user service, packet type (P_{Type})

(ii) Varying queue load (QLoad)

(iii) Bandwidth availability (BW_{avail}),

(iv) Signal intensity (RSSI),

as the input. These inputs act as a vital sign to accurately predict the node's current status in the network. Next, the input variables are fuzzified through fuzzification process to form different Fuzzy Linguistic variables. To fuzzify the variables, fuzzy sets have to be determined. The crisp inputs are classified with its corresponding membership functions to form the fuzzy sets.

In SD_MAC, we have considered 9 fuzzy sets such as

(i) For the HP packet type, the fuzzy sets considered are NSH_{low}, $\text{NSH}_{\text{moderate}}$, and NSH_{high}

(ii) For the MP packet type, the fuzzy sets considered are NSM_{low}, $\text{NSM}_{\text{moderate}}$, and NSM_{high}

(iii) For the LP packet type, the fuzzy sets considered are NSL_{low}, $\text{NSL}_{\text{moderate}}$, and NSL_{high}

where NSH_{high} or NSM_{high} or NSL_{high} indicates the "excellent" state of the node with minimal traffic, less congested HP or MP or LP queues, and low mobility. Similarly, $\text{NSH}_{\text{moderate}}$ or $\text{NSM}_{\text{moderate}}$ or $\text{NSL}_{\text{moderate}}$ indicates "good" state of the node with allowable traffic, moderate congested HP or MP or LP queue and average mobility. Finally, NSH_{low} or NSM_{low} or NSL_{low} indicates "poor" state of the node with high traffic, high mobility and highly congested HP or MP or LP queues, respectively. Threshold values ranges between $\text{TL}_{\text{low,}}$ TL_{med}, and TL_{high} for LP users, $\text{TM}_{\text{low,}}$ $\text{TM}_{\text{med,}}$ and TM_{high} for MP users, and $\text{TH}_{\text{low,}}$ TH_{med}, and TH_{high} for HP users set as per the application scenario. Table 2, describes the classification of inputs and its fuzzy sets.

Fuzzification is followed by the fuzzy rule base creation process which uses logical combination of input variables with the AND (&&) operator. Quality of results in a fuzzy system depends on the fuzzy rules. The sample fuzzy rule base is referred below:

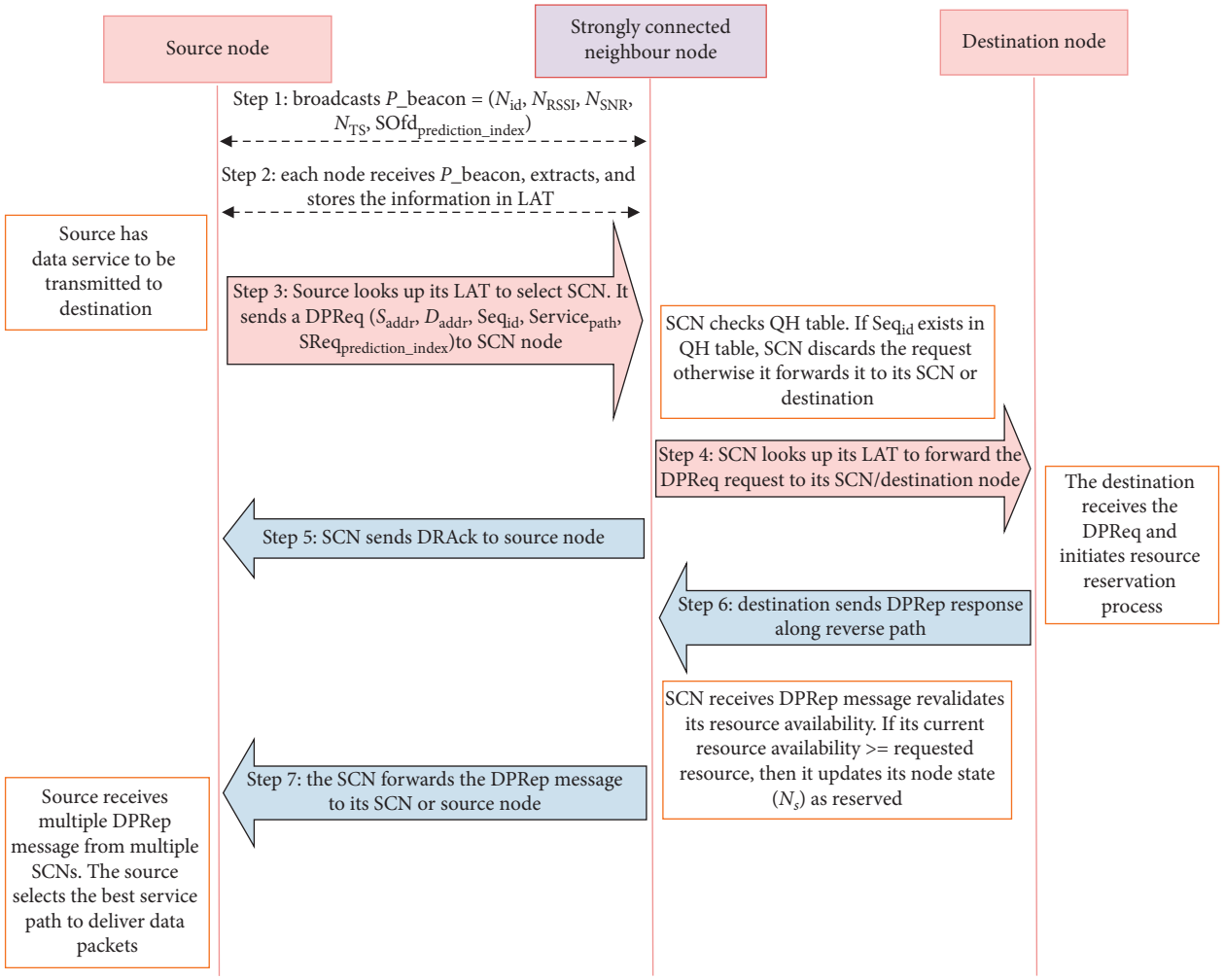

Figure 2: Sequence diagram depicting Routing Service in the SD_MAC model.

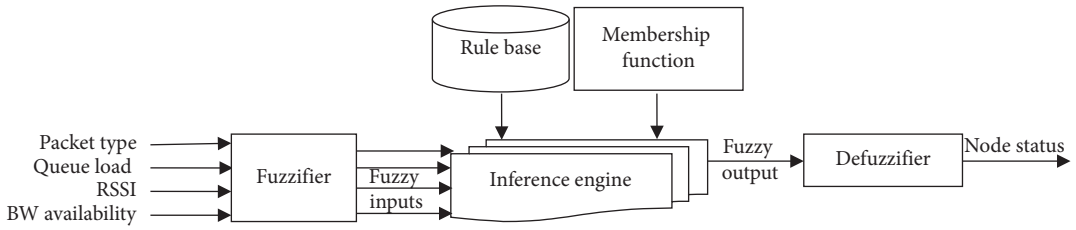

Figure 3: Structure of FIAC process used in NFAC.

If (P_{Type} = HP) && (QL_{HP} = H) && (QL_{MP} = M) && (QL_{LP} = L) && (RSSI = L) && (BW_{avail} = L) then

N_{Status} = NSH_{low}

If (P_{Type} = MP) && (QL_{HP} = L) && (QL_{MP} = H) && (QL_{LP} = L) && (RSSI = H) && (BW_{avail} = H) then

N_{Status} = NSM_{high}

If (P_{Type} = LP) && (QL_{HP} = L) && (QL_{MP} = M) && (QL_{LP} = M) && (RSSI = M) && (BW_{avail} = M) then

N_{Status} = $NSL_{moderate}$

3.4.1. Node Status and Prediction Index Generation. Inference engine uses the rule base to generate the defuzzified value *node status*, N_{status}, which typically reflects the dynamic behaviour of the traffic in node's buffers and the available capacity of the current node. Table 3 displays the N_{status} derived using the QoS parameters.

For example, let us consider the packet type (P_{Type}) be HP, the percentage of queue load for HP be 42% (M), for MP be 35% (L), for LP be 63% (H), the RSSI be 67% (H), and bandwidth available (BW_{avail}) be 28% (L). The triangular membership function representing the range of membership

Table 2: Classification of input "i" and its fuzzy set.

Input field	Range	Values	Fuzzy set
High-profiled user (HP)	$TM_{high} > i < TH_{low}$	$70 > i < 78$	NSH_{high}
	$TH_{low} \geq i < TH_{med}$	$78 \geq i < 85$	$NSH_{moderate}$
	$TH_{med} \geq i \leq TH_{high}$	$85 \geq i \leq 100$	NSH_{low}
Medium-profiled user (MP)	$TL_{high} > i < TM_{low}$	$30 > i < 40$	NSM_{high}
	$TM_{low} \geq i < TM_{med}$	$40 \geq i < 55$	$NSM_{moderate}$
	$TM_{med} \geq i \leq TM_{high}$	$55 \geq i \leq 70$	NSM_{low}
Low-xprofiled user (LP)	$0 \geq i < TL_{low}$	$0 \geq i < 8$	NSL_{high}
	$TL_{low} \geq i < TL_{med}$	$8 \geq i < 18$	$NSL_{moderate}$
	$TL_{med} \geq i \leq TL_{high}$	$18 \geq i \leq 30$	NSL_{low}

Table 3: Crisp output N_{status} derived using the QoS parameter.

P_{Type}	QLoad (%)			RSSI	BW_{avail}	Node status N_{status}
	QL_{HP}	QL_{MP}	QL_{LP}			
HP	H	M	L	L	L	NSH_{low}
	M	M	L	H	H	NSH_{mod}
	L	M	H	H	H	NSH_{high}
MP	L	H	L	H	H	NSMhigh
	M	L	H	M	M	NSMmod
	H	H	L	H	H	NSMlow
LP	H	L	L	M	H	NSLhigh
	M	M	H	L	M	NSLlow
	L	M	M	M	M	NSLmod

degree is represented for each input, as shown in the following figures.

If HP Queue Load = 42%, then the degree of membership ranges between 0.47 and 0.53, as shown in Figure 4.

For MP Queue Load = 35%, then the degree of membership ranges between 0.42 and 0.58, as shown in Figure 5.

For LP Queue Load = 63%, then the degree of membership ranges between 0.46 and 0.56, as shown in Figure 6.

For RSSI = 67%, then the degree of membership ranges between 0.31 and 0.79, as shown in Figure 7.

For bandwidth available = 28%, then the degree of membership ranges between 0.3 and 0.8, as shown in Figure 8.

The fuzzification process combines the input to generate the defuzzified value *node status*, $N_{status} = 79$ (NSH_{mod}), as represented in Table 4. The NSH_{mod} status indicates the node is in good state and can admit allowable data traffic as its queues are moderately congested.

The SD_MAC model considers the N_{status} as one of the primary factor [24] in making decision during admission control. Followed by N_{status}, each node generates a measure of *prediction index*, an estimated value that reflects the node's current resource availability. It is used to validate the current resource utilization of the node based on traffic, mobility, and congestion. The Admission Control module maintains two types of prediction indexes such as

(i) Service-offered prediction index ($SOfd_{prediction\_index}$)

(ii) Service-required prediction index ($SReq_{prediction\_index}$)

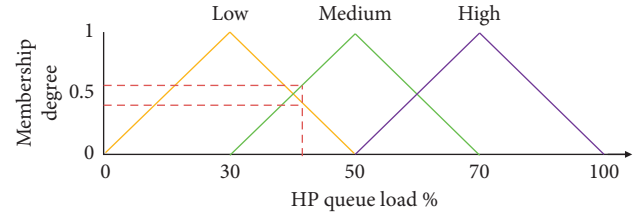

Figure 4: Triangular membership representation for HP Queue Load %.

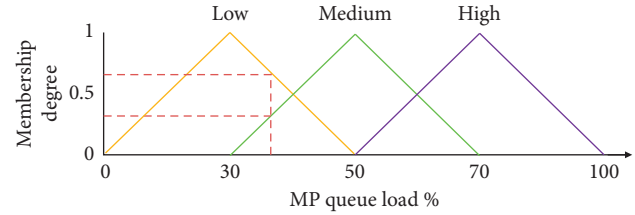

Figure 5: Triangular membership representation for MP Queue Load %.

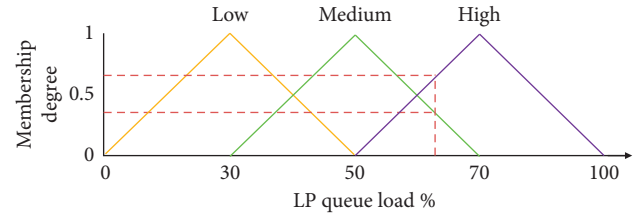

Figure 6: Triangular membership representation for LP Queue load %.

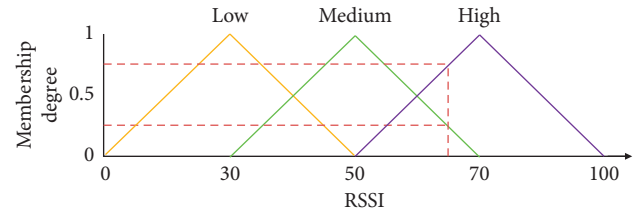

Figure 7: Triangular membership representation for RSSI.

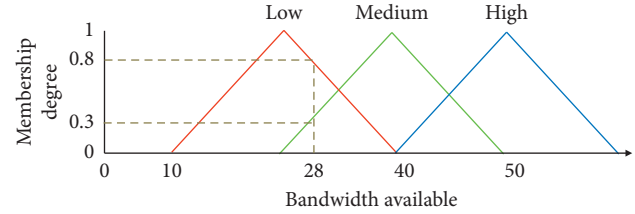

Figure 8: Triangular membership representation for available bandwidth.

Table 4: Crisp output N_{status} derived using the QoS parameter.

P_{Type}	$QLoad$ (%)			RSSI	BW_{avail}	Node status N_{status}
	QL_{HP}	QL_{MP}	QL_{LP}			
HP (0)	42% (M)	35% (L)	63% (L)	67% (H)	28% (L)	79 NSH_{mod}

3.4.2. Service-Offered Prediction Index ($SOfd_{prediction\_index}$). It refers to the service currently offered by the node, i.e., the node's current resource availability is specified using this value. The N_{status} combines with the current load status of HP, MP, and LP queues to generate the $SOfd_{prediction\_index}$. The threshold limits for the HP queue (such as ThQ_{high}, ThQ_{med}, and ThQ_{low}), MP queue (such as TmQ_{high}, TmQ_{med}, and TmQ_{low}), and LP queue (such as TlQ_{high}, TlQ_{med}, and TlQ_{low}) load status are set as per application requirement. Then, $SOfd_{prediction\_index}$ is evaluated by validating the N_{status} along with the queue load status. For example,

if $(N_{status} == NSH_{high})\&\&(HPQ_{load} > ThQ_{high})\&\&(MPQ_{load} > TmQ_{med})\&\&(LPQ_{load} \leq TlQ_{low})$ then

$SOfd_{prediction\_index = V}$; //where, $0 \leq V \leq 1$

end;

where V ranges from $0 \leq V \leq 1$, provided "1" indicates that the node is completely occupied and is unavailable to offer any service currently, while "0" indicates that the node's resources are free and is fully available to offer varied services.

3.4.3. Service-required prediction index ($SReq_{prediction\_index}$). It refers to the service requested by the node, i.e., the node acts as a source and has a *data service* to be transmitted to a destination. The service to be supported by other nodes is specified using this value. The type (TR_{Type}) and measure (M_{Data}) of data service along with the bandwidth required (B_{req}) are validated to generate the $SReq_{prediction\_index}$.

For example,

if $(TR_{Type} == \text{"}_{voice}\text{"})\&\&(M_{Data} > D_{high})\&\&(BW_{req} \leq BWmax)$ then

$SReq_{prediction\_index = V}$; //where, $0 \leq V \leq 1$

end;

where V ranges from $0 \leq V \leq 1$, provided "1" indicates high-profiled data service with high resource requirement is requested by the source, while "0" indicates low-profiled data service with very low resource availability is requested by the source.

Dynamic change in node behaviour (caused due to traffic characteristics, mobility, number of neighbors, etc.) tends to regenerate the dedicated probe request (*prediction index*), which is fed as part of the input to Neuro-Multilayered Learning (NML) to accurately identify strongly connected neighbors. Detailed analysis on the NML process is discussed in the following section.

3.5. Neuro-Multilayered Learning (NML) Process. The proposed work uses the Artificial Neural Network (ANN) to train the model using various input parameters such as the dedicated probe request (prediction index) to accurately predict the strongly connected neighbor nodes. The aim of the proposed model is to recognize the weights of each input parameter using machine learning, specifically, the Artificial Neural Network. The ANN is a supervised mathematical learning tool inspired by biological neural networks. The network is connected with a vast number of nodes called neurons. The connections between the nodes imitate the synaptic connections found in biological entities.

ANNs are incredibly useful for analysis of mapped information as it can be trained to recognize patterns in data.

The aim of implementing the ANN is to predict the weight factors for different nodes such that the best possible node with a weight factor can be selected. The proposed Neuro-Multilayered Learning (NML) mechanism implements a multilayer perception (MLP) to learn, analyze, train, and adapt to the operational behaviour of various components and optimize decision according to the node's current resource availability. The NML maps various features to derive a set of output with good accuracy and high learning capability. It is composed of a vast number of connected units arranged in layers. Among which the first layer is the input layer and the last layer is the output layer. The inner layers are referred to as the hidden layers which are responsible for the bulk of the processing. The process involves important activities such as

(i) Train the network

(ii) Estimate and accurately admit data service

3.5.1. Train the Network. NML initiates a training process to assure adaptive behaviour of the network. Training the network involves estimating the connecting weights by minimizing the overall error measures, i.e., tasks where the weights of each unit (neurons) should be adjusted in such a way that the error between the desired output and the actual output is reduced. The error derivative computed during this process helps to accurately classify each instance. The process involves the following steps:

Step 1: the NML receives input parameters (also referred as instances) including the probe request constituting the $SOfd_{prediction\_index}$ generated from FIAC process. Multiple instances (dynamic change in node's behaviour triggers NFAC engine which derives the $SOfd_{prediction\_index}$ prediction index value) at varied time slot are received by the NML process. Let "i_1" be the instance received at time slot t_1. Change in node's behaviour causes regeneration of an instance "i_2" at time slot t_2.

Step 2: the instances are fed into the NML process to estimate the error derivative using the steps elaborated in "Mathematical Analysis of NML." It then classifies the instances appropriately for accurate decision making. Over a period of time, the training processes craft each node to self-organize and adapt to dynamic changes making the whole network intelligent.

Mathematical Analysis of NML: aim of the NML is to select the best strongly connected neighbor node based on the Credibility Score (CS). The credibility score was calculated as

$$\mathrm{CS} = (\alpha * \mathrm{NN}) + (\beta * \mathrm{DP}) + (\gamma * \mathrm{PF}), \tag{3}$$

where α, β, and γ are the weight factors and NN, DP, and PF are the number of neighbors, dedicated probe request, and packet forward rate factors, respectively.

The proposed model considers 5 parameters as input, such as, number of nodes, average number of neighbors, mobility, dedicated probe request, and packet forward rate. A diagrammatic representation of the artificial neural network of the SD_MAC model is shown in Figure 9.

The input parameters such as number of nodes and mobility determine the kind of network we are dealing with [25], the packet forward rate determines the traffic characteristics, number of neighbors, and dedicated probe request determines the strongly connected neighbor which is why these parameters has been included. The output layer has 8 nodes resulting in 69 possible cases with a combination of α, β, and γ values. The sample values generated for the cases are shown in Table 5.

The cases with 0 values are excluded (the weight factor with 0 relevance would mean that the factor would become irrelevant for the network) to generate 36 possible cases, as depicted in Figure 10, i.e.,

$$\alpha + \beta + \gamma = 1. \tag{4}$$

The NML was implemented to have 5 inputs, 36 outputs, and 3 hidden layers with 10 neurons each. Each node acts as a decentralized processing unit, which, based on its own data activates nodes of the next layer. A node is activated by a weighted sum of all its parent nodes and takes on probabilistic values. For example, let nodes be denoted by the notation a_{ij}, where i refers to the layer corresponding to the node while j refers to the node number. Thus, the first node in the input layer will be named a_{00}, while the last node in the same layer will be named a_{04}. The first node in the next layer will similarly be named a_{10}. The value a_{10} will be a weighted sum of all the values of its parents, viz, a_{00}, a_{01}, . . ., a_{0n}, where n is the number of nodes in the layer. Since the sum will be greater than one, in this work we use a normalizing function ReLU [26] (as ReLU achieves better result than sigmoid [27]) so that the value appears between 0 and 1. The weights are the values assigned to the arrows linking each nodes. We use the notation w_{ijk}, where the arrow connects node j of the layer i to node k of the layer $i+1$. It is by adjusting these weights, that we determine the strength of the connections and how firing of one neuron will activate the next one. To calculate the weights of the zeroth node (a_{10}) on the first layer, and its value is computed by normalizing the sum of all the values in the previous layer multiplied with their weights. Mathematically, in case of node a_{10},

$$\begin{aligned} a_{10} = \sigma\big(&a_{00} * W_{000} + a_{01} * W_{010} + a_{02} * W_{020} \\ &+ a_{03} * W_{030} + a_{04} * W_{040}\big). \end{aligned} \tag{5}$$

For each node, w_{ijk} may be written as w_j. Thus, the equation is rewritten as

$$\begin{aligned} a_{10} = \sigma\big(&a_{00} * W_0 + a_{01} * W_1 + a_{02} * W_2 \\ &+ a_{03} * W_3 + a_{04} * W_4\big). \end{aligned} \tag{6}$$

In general, it is expressed as

$$a_{ij} = \sigma\left(\sum_{j=0}^{n} a_{(i-1)j} \times w_j\right), \tag{7}$$

where a_{ij} is the j^{th} node on i^{th} layer and w_j is the weight of the j^{th} node in the $i-1$ layer. In this manner, all the nodes in the i^{th} layer are assigned values. This continues for all successive layers and the neurons keep firing till finally the output layer is activated. Initially, the model starts with generic weights and in each successive case the weights are refined to better suit the predetermined result. A cost function determines how close the result was to the actual value. This process occurs multiple times till the results converge and weights in each layer obtain the best possible value to suit the output. In our case each of the 36 output nodes was assigned a probability value. The data was divided such that 75% was used for training and 25% for testing. To train our model, we use a simple mean squared error (MSE) as cost function. The MSE is attractive because it is simple to use, parameter-free, and inexpensive to compute. Thus, the MSE is computed as

$$\mathrm{MSE} = \sum_{j=1}^{n} \left(a_{Lj}\right), \tag{8}$$

where L refers to the last layer, n is number of outputs, and "a" is the expected output value. The sum gives us the total error for the network. For example, if we consider only the first four of the 36 possible cases and assume value 0.3 is the correct answer, we may obtain the values, as shown in Table 6, in the first iteration. The error in this scenario would be the sum of all the squares of differences between the observed value and expected value.

$$\begin{aligned} \mathrm{MSE} = &(0.5 - 0.0)^2 + (0.2 - 0.0)^2 \\ &+ (0.3 - 0.0)^2 + (0.4 - 0.0)^2 = 0.94. \end{aligned} \tag{9}$$

Similarly, the MSE is calculated for all possible cases and the average is computed. For each new dataset, consisting of all the weights and biases, the cost function is minimized by reducing the error, and thereby allowing the machine to learn. This model consisting of various parameters becomes capable of predicting the best weight factors (SCN) for the network.

3.5.2. Estimate and Accurately Admit Data Service. Apart from training the network, NML also helps in estimating and accurately admitting data service using the following steps:

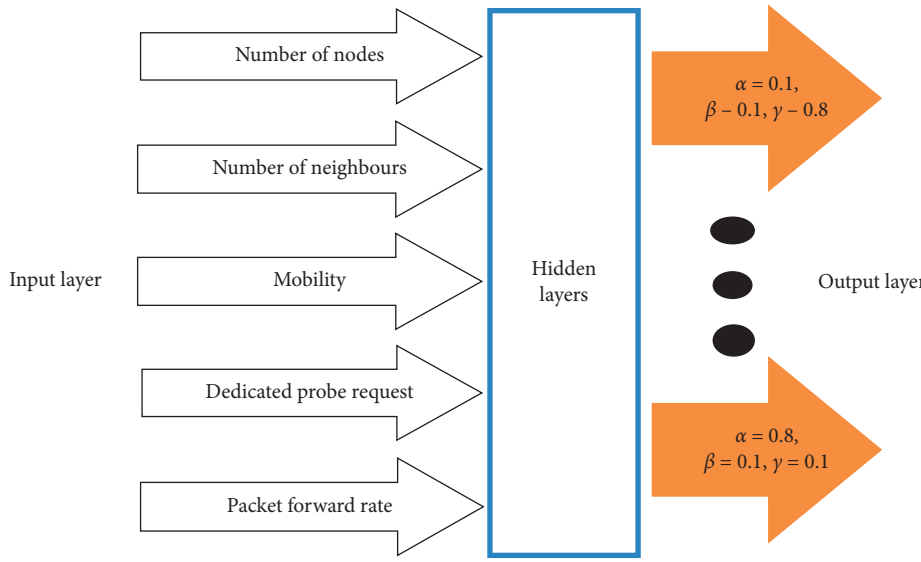

Figure 9: Artificial neural network of the SD_MAC model.

Table 5: Sample of α, β, and γ value including 0 values.

α	β	γ
1	0	0
0	0	1
0.1	0	0.9
0.2	0	0.8
0.3	0	0.7
0.4	0	0.6
0.5	0	0.5
0.6	0	0.4
0.7	0	0.3
0.8	0	0.2

Step 1: at a particular instance of time, each node receives multiple DPReq (S_{addr}, $D_{addr,}$ Seq_{id}, $Service_{path}$, $SReq_{prediction\_index}$) packets from its neighbor nodes. At the same time, the node's AC module maintains its current $SOfd_{prediction\_index}$ value.

Step 2: the node gathers the $SReq_{prediction\_index}$ (instances) from multiple DPReq packets and forwards it to the NML process. First, it categorizes instances based on the traffic type (HP or MP or LP) and groups them together to form a similar set $S = (x_i, y_i, z_i)$, $1 \le i \le n$, consisting of "n" instances, where ($x_i \in$ HP, $Y_i \in$ MP, and $Z_i \in$ LP). Next, it classifies and labels instances either as "Accept candidate-A_C" or "Reject candidate-R_C" or "Unknown candidate-U_C."

(i) Accept candidate, A_C: an instance is said to be labeled as $A_{C,}$ provided the instance satisfies the predefined rule ($SOfd_{predictionindex} > SReq_{predictionindex}$).

(ii) Reject candidate, R_C: an instance is said to be labeled $R_{C,}$ provided the instance satisfies the predefined rule ($SOfd_{predictionindex} < SReq_{predictionindex}$).

(iii) Unknown candidate, U_C: all other instances are labeled as U_C.

The unknown candidate list is fed into NML for further optimization. The multilayered process estimates the error derivative using the steps elaborated in "*Mathematical Analysis of NML*." It then classifies instances with allowable errors as A_C and otherwise as R_C, respectively.

Step 3: the instances ($SReq_{prediction\_index}$) labeled as A_C are prioritized and considered as allowable data service (i.e., the node acts as SCN for its neighbors). The node reserves resources as per the request and triggers the NFAC to regenerate the $SOfd_{predictionindex}$. Repeat the process from step 2.

The labeled candidates are stored into the local Managed Information Database (MID) maintained by each node. The MID has storage capacity for storing varied patterns of labeled candidates. These patterns are then used for data mining and classification during NML's training activity. The idea behind the NML model is to reduce the computational complexity at each level and make the data mining efficient. The unique candidate list from the MID are collected and sent to its neighbors across the network. The neighbor node receives the candidate list and updates its MID with unique candidate list. Algorithm 2 summarizes steps to estimate and accurately admit data service

The error (E_{rr}) between real and expected output is computed. If E_{rr} is within the allowable range (AR), it indicates that instance is covered by an existing attracting basin. Then, θ_{ij} and α_{ij} are adjusted. If E_{rr} is beyond the AR, it means that instance is not covered by any existing attracting basins and need to find "h" with minimum error "u". If E_{rc} is in the AR, it means that the internal output classification represented by unit "u" is applicable to the instance. Also, the input to "j" should be adjusted. Thus, the unit whose AV is the maximum among those connecting with unit "u" is selected. If E_{rc} is beyond the AR, it means that both input classification and output classification is inadequate for the instance. Thus, "j" and "h" are appended to the hidden layers. The new second layer "s" is connected with

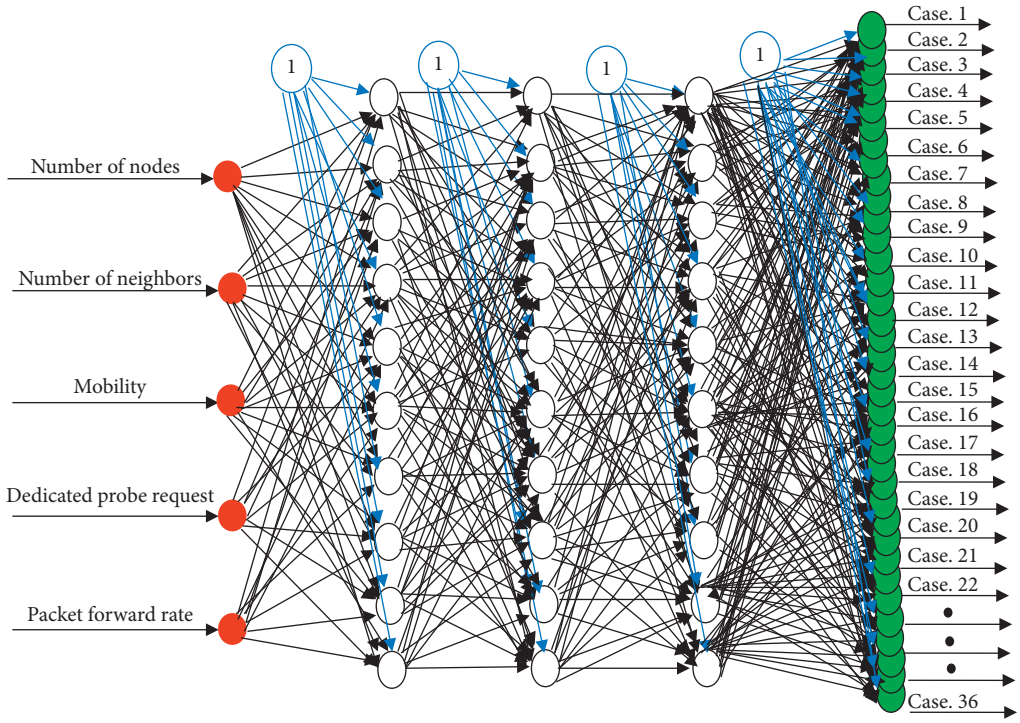

Figure 10: NML formed in SD_MAC with input, hidden, and output layers.

Table 6: Sample values generated in first iteration.

α	β	γ	Observed value	Expected value
0.1	0.1	0.8	0.3	0
0.1	0.2	0.7	0.2	1
0.1	0.3	0.6	0.9	0
0.1	0.4	0.5	0.4	0

all the input units. The new third-layer unit is connected with all the output units to generate a fine tuned expected output. The error derivative analysis using the internal layers of the NML process is represented in Algorithm 3.

The NML process considers the error $(E_{rr})(E_{rc})$ value which is the difference between the real and expected output, to classify the instances as follows:

(i) If error is within allowable range, then they are classified as "Accept candidate, AC"

(ii) If error is beyond allowable range, then they are classified as "Reject candidate, RC"

After several execution of the training stage, NML starts testing operation to test the correctness of the results. The success of network learning is to satisfy the user data flow requirement without performing any operations at the node level. This makes the entire network to offer data service accurately and select the most suitable SCN nodes.

3.6. Classifier, Traffic Regulator, and Scheduler. Any given network supports various types of data flows and categorizes it into two types:

(i) *nonreal-time traffic or best-effort flows* such as messages which requires the data to be reliably delivered to the destination

(ii) *real-time traffic or* QoS *flows* such as voice and video which apart from reliability requires additional constraints such as available bandwidth and delay to be satisfied.

The real-time traffic is normally expected to get better service than the others. Hence, priority for acquiring resources in a network is generally given to the real-time traffic. In addition to real-time flows, nonreal-time flows such as messages play a very crucial role because of their application in military battle field and emergency rescue operations. These applications require fast and assured transmission of emergency messages. Hence, in such a scenario, priority for acquiring resources in a network has to be given to the emergency nonreal-time traffic based on the user profile. QoS in MANET becomes a necessity due to its applications in critical scenarios. The SD_MAC model makes use of the *classifier*, *traffic regulator*, and the *scheduler* modules to distinguish, regulate, and schedule data traffic appropriately. Figure 11 shows how these modules work

```
Input: The instances i1, i2, i3 received from Neighbor nodes
Ī = generate_set (i1, i2, i3); // Ī denotes the multi-instance sets
Initialize icount = 0; j = 1;
icount = getCount(I); // get the number of instances (icount)
for j = 1 to icount
repeat // repeat and validate multi-instance set Xi, Yi
   if (ij == Xi) then
      if (SOfdpredictionindex > SReqpredictionindex) then
         AC_list = Ii
            end;
   else if (ij == Yi) then
      if (SOfdpredictionindex < SReqpredictionindex) then
         RC_list = Ii; //Instance labeled as reject candidate
      end;
   else
      UC_list = Ii; //instance labeled unknown candidate
   err = derive_Error(); // find error derivative
   calssify_candidates(err); //classify instance AC or RC or UC
   end
j = j+1;
until (j < icount); // continue for all instances
end / * end of for loop
M label = getMasterCandidate(AC_list, RC_list, UC_list);
add_to_MID(Mlabel) //add master list to DB.
send_to_neighbor( Mlabel)
Output: Classify instances and generate error derivative
```

Algorithm 2: Estimating and admitting data service.

```
Err = DeriveError (Routput, Eoutput); // error Err
if (Err <= Alwrange) then // Alwrange is allowable range
   Alwvalue = AdjustSlayer (θij, αij); //θij and αij are adjusted.
   Ii = classifyCandidate (Err, CInstance, Avalue);
   ACandidate_list = Ii // instance is set as acceptable candidate
   updateMID (ACandidate_list);
else if (Err > Alwrange) then
Tlayer = findMerrorTLUnit(); //find 3rd-layer min error
Erc = ComputeCharacteristicerror (sendtoTlayer(CInstance));
   if (Erc <= Alwrange) then
   Ii = classifyCandidate (Erc, CInstance);
   ACandidate_list = Ii
   updateMID (ACandidate_list); //Assigned as accept candidate
   else if ( Err > Alwrange) then
   RC = IOCinadequate(Ilist);
   if (RC < Thvalue) then
      Ii = classifyCandidate (Erc, CInstance);
      ACandidate_list = Ii;
      updateMID (ACandidate_list);
   else
      Ii = classifyCandidate (Erc, CInstance);
      RCandidate_list = Ii; //Assigned as reject candidate
      updateMID (RCandidate_list);
   end;
end;
else
Ilist = addinformation (CInstance, Err, Erc);
end;
```

Algorithm 3: The error derivative analysis of the NML process.

together to provide the service. The classifier, scheduler, and traffic regulator modules operate between the IP layer and the MAC layer.

In SD_MAC model, after the source selects the best service path (a path that is established for an acceptable period for transmission with intermediate SCNs that have the capability to serve source node's QoS requirement) to deliver the data traffic, it then admits data traffic along the selected path to the destination. At each node, the admission control module maintains a $\mathrm{SOfd}_{\mathrm{predictionindex}}$ to reflect the node's current resource availability (in MANET the node's current resource utilization keeps changing due to change in traffic, mobility, and congestion) using which it revalidates if the $\mathrm{SOfd}_{\mathrm{predictionindex}} > \mathrm{SReq}_{\mathrm{predictionindex}}$ (indicates the data service to be supported by intermediate nodes). If the $\mathrm{SOfd}_{\mathrm{predictionindex}}$ is high, then the admission controller admits the session for processing otherwise rejects it. When a new session is admitted, the classifier, the traffic regulator, and the scheduler performs the following process to support the desired QoS. Based on the traffic type, the *classifier* classifies the packets QoS profile as platinum, gold, silver, and bronze. It then marks the priority of those packets HP (High Profiled, code is 0), MP (Medium Profiled, code is 1), and LP (Low Profiled, code is 2). Thus, the classifier differentiates real-time and nonreal-time sessions forcing the *traffic regulator* to process MP and LP packets but not the HP packets. For example, if the traffic type is "network control," then the QoS profile is set to "Platinum," and the user priority is marked as HP. Then, based on marking, the packets remain unregulated and bypass the traffic regulator module if it is HP and are forced to pass through the traffic regulator if they are MP or LP. The goal of the traffic regulator is to regulate the data traffic of MP and LP packets (i.e., it tends to delay MP and LP packet processing) and utilize the resource for processing HP packets (real-time sessions). The stamped (marked) packets are then queued by the scheduler in their respective HP, MP, and LP queues and wait for transmission. The *scheduler* implements a combination of Strict and Probabilistic Priority Queuing to impose service differentiation as per the user's priority and serve packet in their respective HP, MP, and LP queues. It permits users to utilize resources opportunistically by regulating high traffic imposed by other nodes.

Detailed specification on classifier, traffic regulator, and scheduler modules are referred in the following section:

3.6.1. Classifier. To meet the user's service requirements and provide better service to selected network traffic, the *classifier* module in SD_MAC differentiates packets into four classes: platinum, gold, silver, and bronze, as referred in Table 7.

Based on the type of traffic, the packets are stamped as

(i) High-Profiled user (HP) supports high bandwidth to ensure high quality of service

(ii) Medium-Profiled user (MP) supports nominal bandwidth for moderate QoS

(iii) Low-Profiled user (LP) supports lowest bandwidth for bulk data service

The first class "platinum" and "gold" has the highest priority and corresponds to applications with real-time (RT) traffic such as voice and network control packets. This class supports high bandwidth to ensure high QoS. The class is referred to as "expedited forwarding" and in IntServ is referred as guaranteed service. The second class "silver" has less priority than the first class. It is suitable for applications requiring high throughput such as video applications. This class supports nominal bandwidth for moderate QoS. The class is referred to as "assured forwarding" in DiffServ and "controlled load" in IntServ. The third class "bronze" is the least priority class and has no specific constraint. This class supports lowest bandwidth for bulk data service. This class is referred to as best effort in both DiffServ and IntServ architectures.

Based on Traffic type and QoS profile, each user is assigned a static priority as per the classification HP, MP, and LP. To meet service requirements, a proportional weight (which favours fairness among the competing nodes) is assigned to each user. This is implemented by adding a priority field to the header of every packet. The codes (0, 1, and 2) representing the priorities are stored in the priority field. The intermediate node does enqueuing based on the priority value.

3.6.2. Traffic Regulator. The *Traffic Regulator* module regulates MP and LP (best effort) traffic by delaying best-effort packet processing and allows more bandwidth for HP (real-time) packet processing. When a new real-time flow is allowed by the admission control block, the packets marked as HP by passes the traffic regulator and are queued in HP queues for processing.

3.6.3. Scheduler. The *Scheduler* module employs a new scheduling discipline referred to as the Strict and Probabilistic Priority Queuing (SPPQ) which determines the probability with which the queue is served by assigning a parameter to each priority queue. It offers variety of service differentiation among various groups of traffic classes. To provide high QoS, different priorities are assigned to packets entering the nodes in the network. The packet scheduling algorithm is employed to select each packet and transmit it to the data link layer. Though a number of packet scheduling algorithm have been proposed, the prominent ones among them include Strict Priority (SP) (i.e., servicing lower priority packets only when higher-priority ones are not waiting to be serviced) [28], Weighted Fair Queuing (WFQ) (i.e., having N data flows currently active, with weights $w_1, w_2, \ldots, w_n$, data flow i will achieve an average data rate of $R * w_i / (w_1 + w_2 + \cdots + w_N)$, where R is the data link rate) [29], and Weighted Round Robin (WRR) (i.e., assigning a portion of the available bandwidth to each priority queue) [30] are perhaps the most widely adopted disciplines. Among the three disciplines, the SP is easy to implement and provides large differentiation among classes. However, the SP is

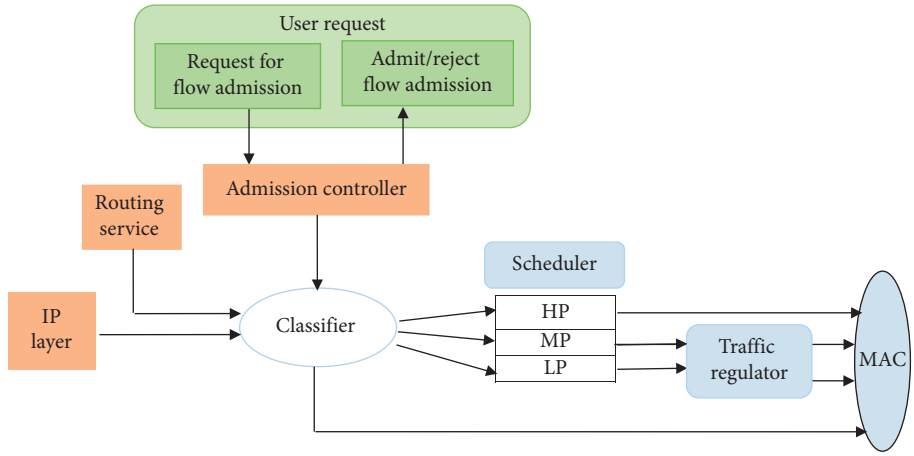

Figure 11: SD_MAC model with classifier, traffic regulator, and scheduler modules.

Table 7: QoS Profile supported in the SD_MAC model.

Traffic type (TR_{Type})	QoS profile	User priority (P_{Type})	Codes representing priority
Network control	Platinum	High profiled	0
Voice	Platinum	High profiled	0
Interactive video	Gold	High profiled	0
Mission critical	Gold	High profiled	0
Best effort	Silver	Medium profiled	1
Bulk data	Bronze	Low profiled	2
Scavenger	Bronze	Low profiled	2

unfair to all classes other than the highest priority class. This mechanism of SP may introduce large periods of starvation to lower priority classes and usually cannot handle this starvation problem by itself. To deal with starvation problem and make the SP discipline controllable, we propose to assign a parameter to each priority queue, which determines the probability with which the queue is served. Hence, a new scheduling discipline, referred to as the Strict and Probabilistic Priority Queuing (SPPQ) is formed. Similar to WFQ and WRR, variety of service differentiation can be achieved in SPPQ by setting the assigned parameters properly. The novel property of the SPPQ discipline is that it provides service segregation among groups of traffic classes and has the following characteristics:

(a) It is easy to implement and verify

(b) It exploits well the available network bandwidth

(c) It consumes very small memory and processing power requirements

(d) Network administrators find them easy to understand and configure

(e) It does not require time-stamping as in WFQ

The performance of SPPQ algorithm has been evaluated in this work by considering different priority classes and traffic patterns. The evaluation shows that with high network load i.e., the packet arrival probability $\lambda > 0.6$, the QoS offered to low-priority packets rapidly deteriorates with its throughput significantly dropping and delay sharply increasing.

In this paper, we consider the network to operate under the following assumptions:

(i) The arrival process of each input of the network is a simple Bernoulli process, that is, the probability that a packet arrives within a clock cycle is constant and the arrivals are independent of each other. We will denote this probability as λ. This probability can be further broken down to λ_h, λ_m, and λ_l, which represent the arrival probability for high- , medium- , and low-priority packets, respectively. It holds that $\lambda = \lambda_h + \lambda_m + \lambda_l$.

(ii) Under this mechanism, when applications transmit a packet to the network, they specify its priority, designating it either as high, medium, or low. The criteria for priority selection may stem from the nature of packet data.

(iii) Each packet priority queue is statically assigned a *weight*, which specifies the bandwidth ratio that will be dedicated to the particular queue with their sum must be equal to 1.

(iv) Upon reception, packets are first classified according to their priority and are then assigned to the queue specifically dedicated to the particular priority, as depicted in Figure 12.

During the network cycle, the SPPQ algorithm evaluates the queues by observing the bandwidth ratio that has been assigned to each queue to select the packet to be forwarded through the output link. The packet is always transmitted

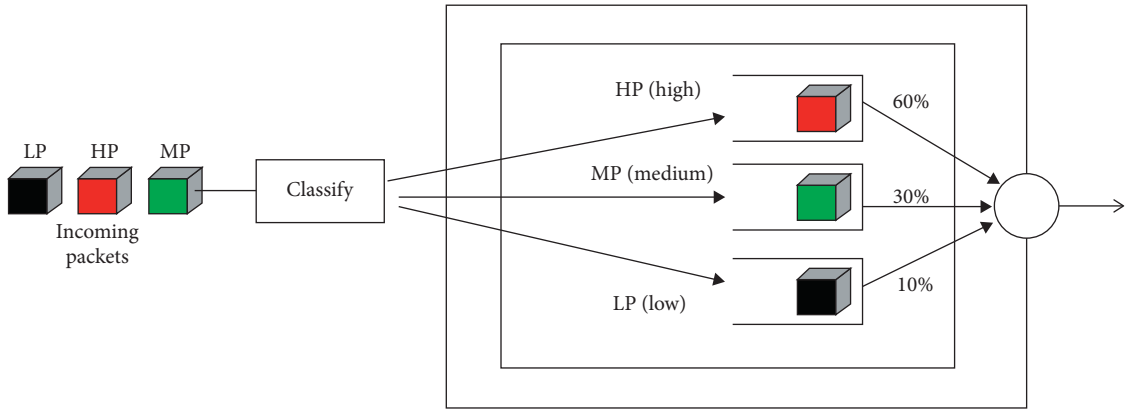

Figure 12: The SPPQ mechanism of the SD_MAC model.

when there is traffic waiting, as opposed to nonwork conserving algorithms which do not transmit a packet if the queue is found to be empty. The bandwidth is divided among the queues that do have packets to transmit, proportionally to their weights in case if the queue does not use its bandwidth ratio within a time window.

Every node maintains a separate queue such as HP, MP, and LP for the four classes of users.

According to the priority, the packets are enqueued and dequeued from the queues proportionally based on the weights and the percentage of packets waiting in their respective queues. Let a, b, and c represent the percentage of HP, MP, and LP packets waiting in their respective queues and it is calculated as follows:

$$a = \frac{QL_{HP}}{(QL_{HP} + QL_{MP} + QL_{LP})},$$

$$b = \frac{QL_{MP}}{(QL_{HP} + QL_{MP} + QL_{LP})}, \quad (10)$$

$$c = \frac{QL_{LP}}{(QL_{HP} + QL_{MP} + QL_{LP})},$$

where QL_{HP}, QL_{MP}, and QL_{LP} represent the number of packets occupied in HP, MP, and LP queues and is dynamic. From each queue, the number of packets to be dequeued is decided based on their access ratio and is calculated as follows:

$$w_{HP}a : w_{MP}b : w_{LP}c, \quad (11)$$

where "w" represent the weight which is a constant. The w_{HP}, w_{MP}, and w_{LP} are the user defined weights assigned for HP, MP, and LP queues such that $w_{max} > w_{HP} > w_{MP} > w_{LP} > 0$. The w_{max} represents the maximum weight. Thus, at any point of time, for any random data flow, the priority of HP packets over MP and LP and the priority of MP packets over LP is ensured.

4. Simulation Results and Discussion

We compare in this section the performance of our proposed SD_MAC scheme with other well-known admission control protocols such as SWAN, StAC, DACME, and FuzzyQoS. For evaluation purpose, the simulator was developed using MATLAB. The simulation prototype provides a user-friendly Graphical User Interface (GUI) enabling the user to edit the network topology and set the parameter values of the protocol module in just a few mouse clicks.

The GUI of SD_MAC is shown in Figure 13.

The simulation environment was tested with 50 nodes moving in an area of 1000×1000 sq m. Each node implements random waypoint mobility model. The transmission rate of nodes is 2 Mbps, while their transmission range is 250 m. Initially, when the simulation is initiated the nodes are deployed at random location in the geographical area, as shown in Figure 14.

The mobile nodes move randomly and freely with a speed varying between 0 m/s to 35 m/s. The network traffic model is the Poisson distribution model with buffer space in each node configured as 100 packets. The application with mixture of traffics such as FTP, Web microflows, VoIP, and real-time traffics. The voice and video flows represent real-time traffic with voice modelled as 32 kbps constant rate traffic with a packet size of 80 bytes, video modelled as 200 kbps constant rate traffic with a packet size of 512 bytes, and TCP flows are greedy FTP traffic type with packet size of 512 bytes are used during evaluation.

The parameters used as part of simulation are listed in Table 8.

During simulation, the nodes are randomly chosen to act as the source (node that generates message and transmits the message to destination), with 20 sessions/source. The source and destination nodes are distributed among the mobile nodes in the MANET. Figure 15 displays the network model of the SD_MAC system.

To better investigate the scope and performance [31–35] of the protocols, the metrics are selected carefully for fair result analysis. With constant node density scenario, by varying the speed of nodes (as it affects the performance of the protocols due to frequent topology changes) the simulation was run for 500 iteration to derive the average results and the observation on various metrics is analyzed.

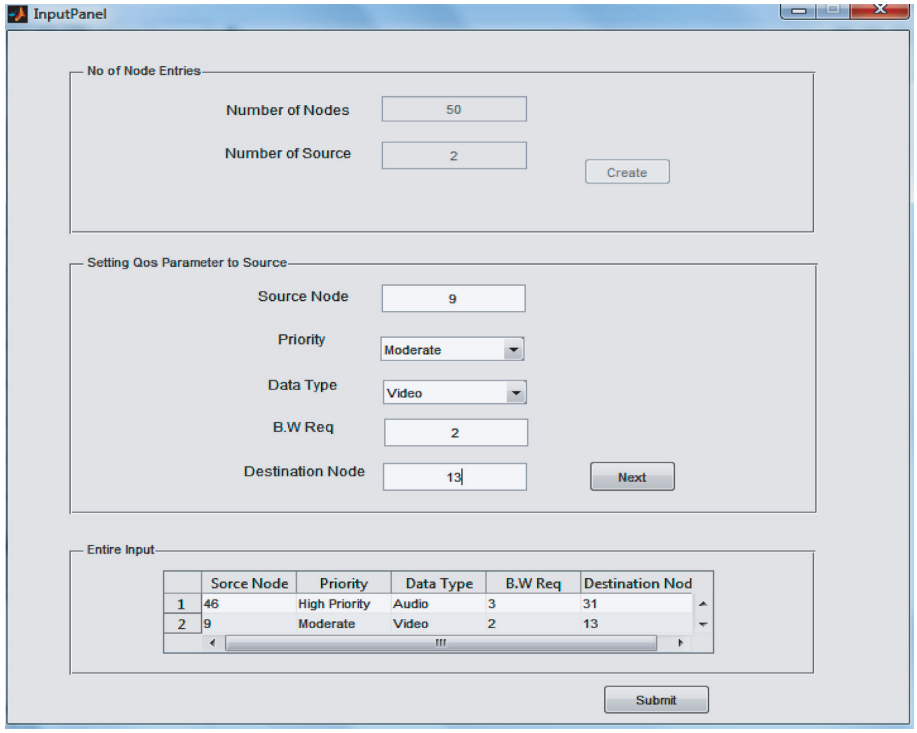

Figure 13: GUI of the SD_MAC model.

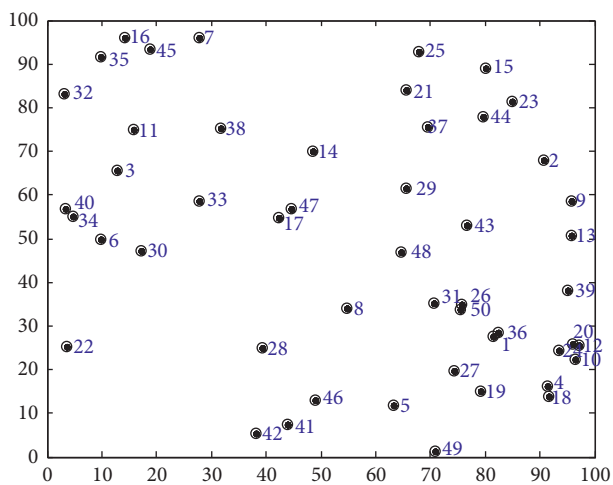

Figure 14: Network model with nodes deployed at Random Location.

Scenario 1: in this scenario, the number of nodes is varied from 50 to 300 while the nodes were set to move with low mobility of 5 m/s.

Scenario 2: in this scenario, the number of nodes is varied from 50 to 300 while the nodes were set to move with high mobility of 30 m/s.

4.1. Packet Delivery Ratio (PDR). PDR defines the ratio of the number of packets delivered to the destination successfully against the total number of packets generated by the source.

Table 8: Simulation parameters.

Parameters	Values
Simulator	MATLAB
Simulation time	100 s
Topology area	1000 m × 1000 m
Number of nodes	50–300
Mobility speed	0 to 35 m/s
Mobility model	Random waypoint model
Transmission range	250 m
Transmission rate	2 Mbps
Packet size	80 and 512 bytes
Traffic type	Constant bit rate (CBR)
Number of CBR connections	5
Pause time	0 s
Channel type	Wireless channel
Antenna type	Omni antenna
MAC type	IEEE 802.11
Frequency	2.4 GHz

For better insight and comparative analysis among various approaches, simulation results for scenario 1 and scenario 2 are captured and presented in Tables 9 and 10.

4.1.1. Scenario 1 Analysis. In scenario 1, all the nodes exhibit low mobility across the network. The PDR data captured for various approaches are represented in Figure 16(a). From Figure 16(a), we observe that though the FuzzyQoS scheme performs better in delivering packets successfully to destination when compared to DACME, StAC, and SWAN, the proposed SD_MAC scheme demonstrates better PDR than FuzzyQoS. The fact

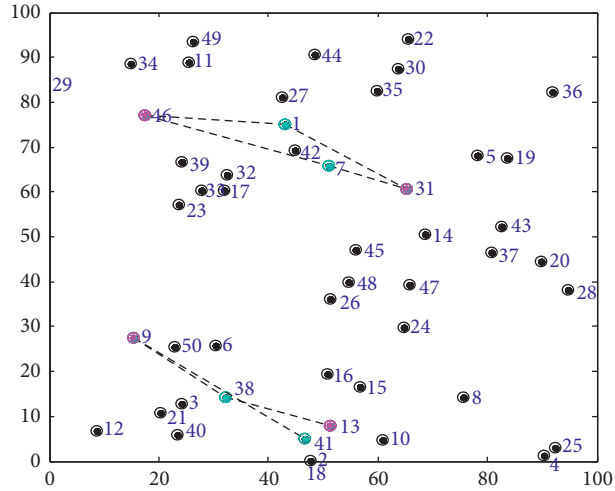

Figure 15: Network model of the SD_MAC System.

Table 9: Scenario 1: nodes varied from 50 to 300 with mobility of nodes 5 m/s.

	Scenario 1: packet delivery ratio (%)										
	Nodes 50	Nodes 75	Nodes 100	Nodes 125	Nodes 150	Nodes 175	Nodes 200	Nodes 225	Nodes 250	Nodes 275	Nodes 300
SWAN	69	76	79	88.5	89.5	91.2	93.5	95.5	96.5	97.7	99
StAC	69	75	80	88.5	91.5	92.2	94.5	97.0	97.5	98.7	99.1
DACME	70	78	86	89.5	89.5	91.2	94.5	96.9	97.5	98.7	99.1
FuzzyQoS	75	88	91	95	96	97	98	98.5	98.9	99.6	99.8
SD_MAC	88	94	97	97.5	98.5	98.5	98.7	99.1	99.2	99.8	99.9

Table 10: Scenario 2: nodes varied from 50 to 300 with mobility of nodes 30 m/s.

	Scenario 2: packet delivery ratio (%)										
	Nodes 50	Nodes 75	Nodes 100	Nodes 125	Nodes 150	Nodes 175	Nodes 200	Nodes 225	Nodes 250	Nodes 275	Nodes 300
SWAN	64	72	76	84.5	83.5	85.2	87.5	87.9	89.5	91.7	93
StAC	66	70	74	84.5	85.5	86.2	87.5	87.9	90.5	91.7	92.3
DACME	69	72	76	86.5	87.5	88.2	88.5	88.9	91.5	93.7	94.1
FuzzyQoS	70	80	90	94	95	96	97	98	98.5	99.4	99.7
SD_MAC	82	92	96	96.5	97.5	98.2	98.5	98.9	99.5	99.7	99.8

that in the SD_MAC model, the packets are admitted by the admission controller through the SCN nodes delivered successfully to the destination. The path detected remains intact due to low mobility. Significantly it is observed that when the node density increases, the percentage of PDR also increases because the number of SCNs identified during path selection is more enabling SD_MAC to select the best among them resulting in high PDR. With 100 nodes, SD_MAC surpassed DACME's, StAC's, and SWAN's performance by ~10%, ~15%, and ~18% and FuzzyQoS by ~5%–~8%. Nevertheless, even when the number of nodes is increased to 300, the proposed scheme was able to sustain the PDR% > 95% and remain stable in delivering packets successfully to destination when compared to other schemes.

4.1.2. Scenario 2 Analysis. In scenario 2, all the nodes exhibit high mobility across the network. In Figure 16(b), we can observe the PDR of the proposed SD_MAC scheme outperforms other schemes and maintains the PDR above 80% even when the node density increases and exhibits high mobility. The Neurofuzzy Inference-based Admission Control engine in the SD_MAC model exploits the linguistic control capabilities of the fuzzy logic controller and the learning capabilities of the neural networks to handle incoming traffic and utilize the resources at an optimal level. It is intelligent and fast detection of SCNs helps in achieving high PDR even when mobility increases. Additionally, the neuro-multilayered learning technique ensures the network to adapt to dynamic behaviour and make accurate decisions spontaneously. Though DACME has better PDR compared to StAC and SWAN, and its PDR starts degrading when the number of nodes increases. From the result it is observed that when the node density is 125, the SD_MAC surpassed SWAN's and StAC's performance by ~12%–~15% and DACME by ~10%. The PDR for SD_MAC is

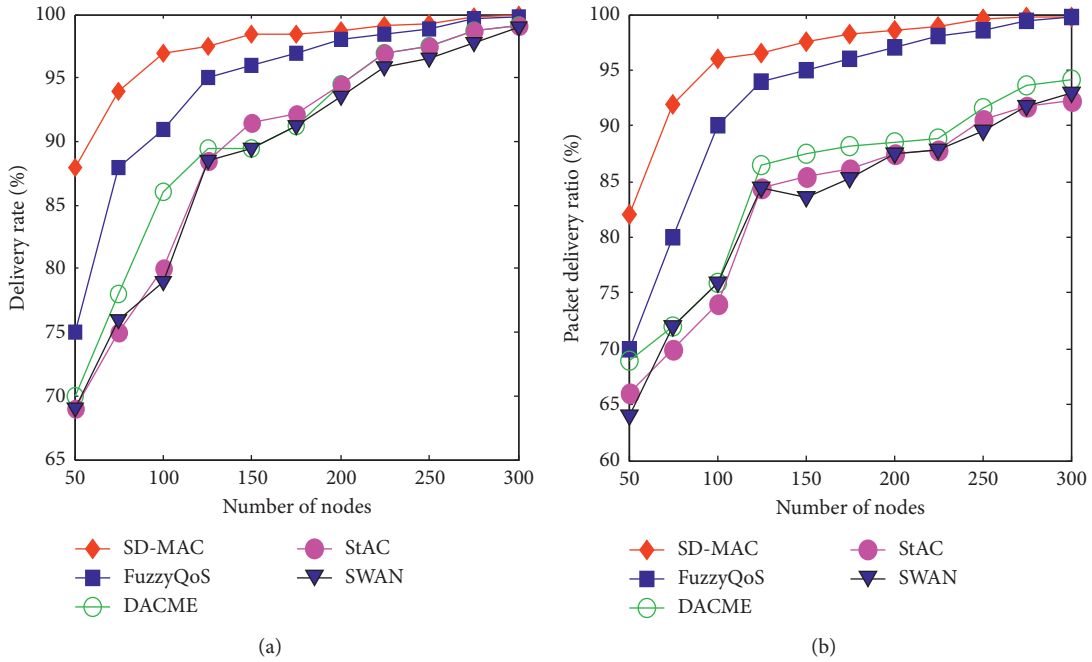

Figure 16: (a) Nodes vs PDR with node speed 5 m/s. (b) Nodes vs PDR with node speed 30 m/s.

found to be ~3%–~5% more compared to FuzzyQoS. Moreover, the PDR is sustained above 90% when the number of node increases above 125. It has been noticed that even with dynamic change in topology (considering scenario 1 and scenario 2), the nodes were able to successfully deliver the packets from source to destination and sustain high percentage of PDR compared to other existing counterparts. The fact that the LAT look-up process performed by each node during path selection resulted in identifying one or more SCNs preventing the node from waiting for the time slot in order to perform a LAT look up once again. It is from the PDR observation that the path selected was successfully utilized for data transmission in the proposed model, making it most suitable scheme for critical applications.

4.2. Session Admission Ratio (SAR). It refers to the ratio of data sessions admitted into the network to the total number requesting admission. Figure 17 shows the SAR of the protocols at different node speed.

SAR for StAC is high compared to others as increase in node speed impacts earlier admitted sessions making source to admit more sessions. Even with high-node speed, SD_MAC surpassed FuzzyQoS performance on an average by ~18%–~22%, DACME and SWAN by ~31% and ~35%, respectively. Though FuzzyQoS uses explicit notification to make the source aware of any sudden change in topology, it does not handle well the route failures, making it to generate more routing overheads to recover the data route.

While, DACME does periodic thorough assessment of resources to identify path conditions during session admission, change in conditions requires reassessment to assure fairness in this approach degrading its SAR. Even in high node speed occasions, SD_MAC is able to sustain SAR

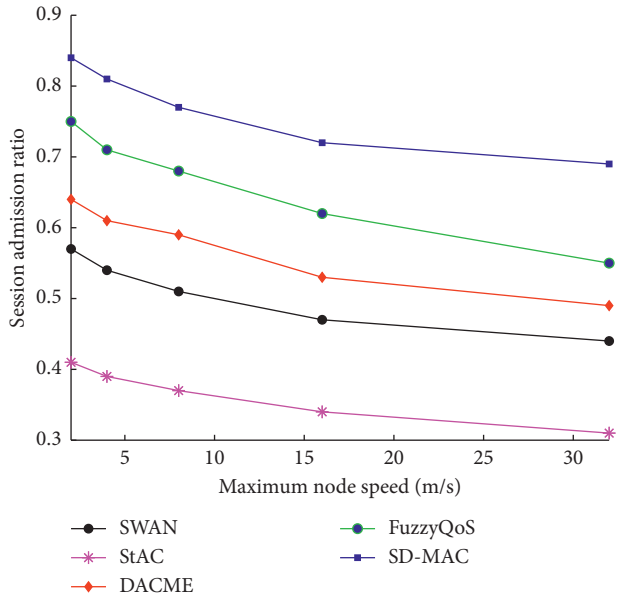

Figure 17: Session admission ratio for varying node speed.

and successfully serve the admitted data sessions against other approaches. The fact being each node in SD_MAC has the capability to select its SCNs by LAT look up and share among each other the type of services offered. During probe request, it selects SCNs instantaneously reducing the overall time taken during the probing session making it fast and successful.

4.3. Session Completion Ratio (SCR). It refers to the ratio of the number of data sessions completed to the application's satisfaction. Figure 18 represents the Session Completion

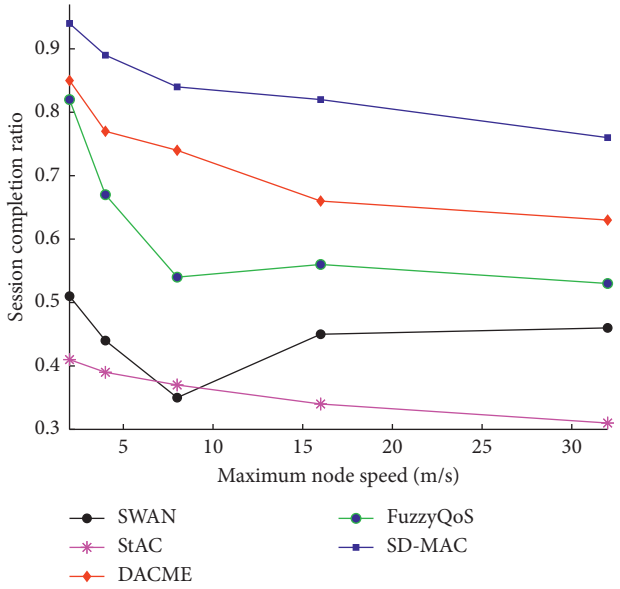

Figure 18: Session completion ratio for varying node speed.

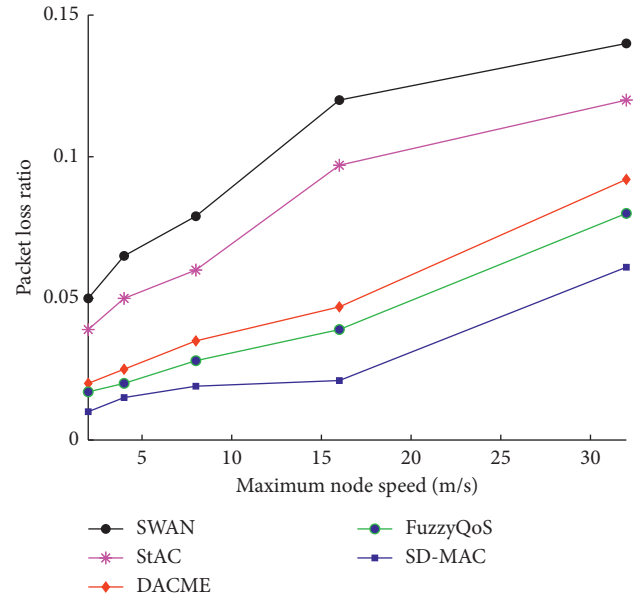

Figure 19: Packet loss ratio vs node speed.

Ratio (SCR) of the protocols. The excessive retransmission results in switching the data session to another route or initiate new route discovery.

The session drops if its requirements are not fulfilled. It is clear from figure that the SCR of SD_MAC varies from 94.3% to 76.8% by increasing speed from 2 m/s to 32 m/s while for StAC it varies from 41.2% to 31.4%, respectively. In case of StAC, admitted sessions are dropped due to link failures affecting earlier admitted session resulting in frequent route discoveries. SCR of DACME is higher than FuzzyQos even at higher node speed because it takes advantage of periodic assessment of end-to-end conditions during path probing and admitting traffic only when the conditions are appropriate assuring high SCR for sessions admitted. SCR of FuzzyQos is low compared to DACME and SD_MAC, as it does not deal well with route failures while higher speed cause frequent route failure making SCR to degrade. The SD_MAC performs better at different node speed among all the protocols. It is fast, intelligent, and controlled admission of data session helps in achieving high SCR. Obviously, the neuro-multilayered learning process makes the whole network learn and adapt to dynamic node behaviour, making the system to make accurate decisions spontaneously. Local route repair carried out through LAT look up helps to maintain the agreed throughput to the data session.

4.4. Packet Loss Ratio (PLR). It refers to the ratio of the number of lost packets to the total number of sent packets.

In Figure 19, it is observed that compared to other approaches, the packet loss ratio for SWAN mechanism is higher and increase from 1.5% to 10.7% when node speed changes from 2 m/s to 32 m/s. Higher node speed makes the preprobed routes stale in SWAN resulting in high packet loss. PLR for FuzzyQoS is low compared to SWAN, StAC, and DACME, as it incorporates methodology to adapt to traffic transmission rate under dynamic conditions which reduces data rate for affected session and admits it in new routes.

Observation shows that SD_MAC has the lowest PLR and ranges from 0.15% to 0.61% when node speed changes from 2 m/s to 32 m/s. The multilayered neurofuzzy-incorporated AC in SD_MAC switches data session from one route to another when current route fails. The capability to spontaneously identify SCNs and predict the node's current resource availability makes the system to admit sessions faster and on more appropriate routes. It not only increases SCR in SD_MAC but also reduces the overall PLR, respectively.

4.5. Average End-to-End Delay. It refers to the time slot elapsed between the time at which the packet is generated at its source and the time it is delivered to its destination. Figure 20 shows the average end-to-end delay caused with increase in node speed for various approaches.

Increase in node speed deteriorates the performance of link causing interference resulting in packet loss, route failures, or delayed transmission. Route failures in StAC make the source to reject the affected session and readmit the session via the backup routes. Frequent route failures not only initiates route discovery frequently but also introduces more overheads and delay which is the case indentified in StAC. The observation indicates the proposed SD_MAC system's efficiency in sustaining low delay making it more suitable for real-time applications. Prior to data traffic admission, the NFAC engine enables accuracy in decision making, i.e., selecting SCNs, allocating, and utilization of available resources not only assures successful guaranteed session admission but also reduces the overall end-to-end delay.

4.6. Average Throughput. It refers to the total amount of data successfully delivered to the destination per unit time.

In most approaches, frequent route-failures caused due to high mobility makes nodes fail to deliver packets successfully to destination. In Figure 21, the throughput is high

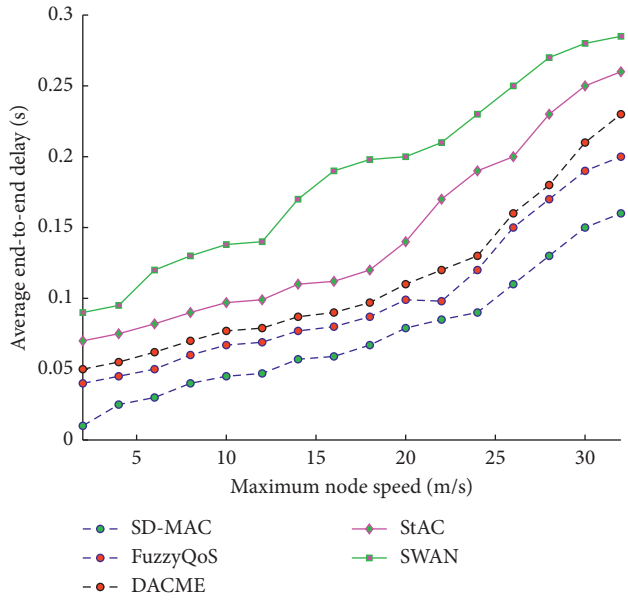

Figure 20: Average end-to-end delay Vs node speed.

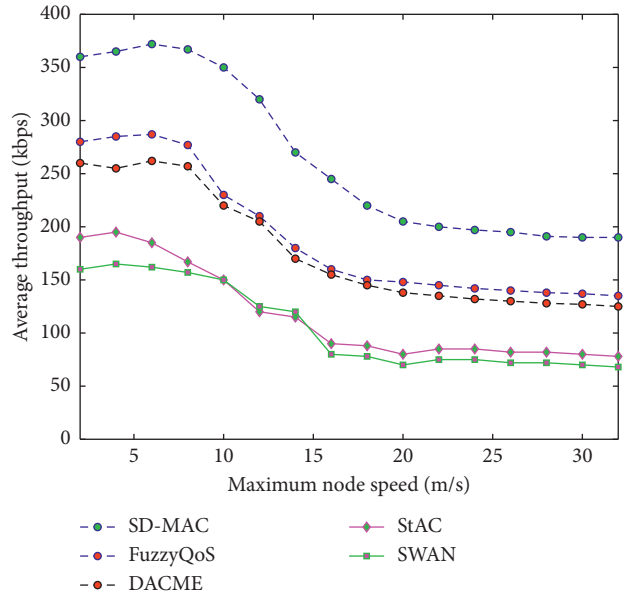

Figure 21: Average throughput vs node speed.

when node speed (NS) is low (NS < 10 m/s). Optimal amount of throughput is achieved for all approaches when NS is between 2 m/s to 10 m/s. The throughput for all approaches drops when the node speed varies from ~10 m/s–~15 m/s and then upholds.

Observation in Figure 21 shows that SD_MAC has maintained its highest average throughput due to its highest SCR as it is able to assure and uphold the throughput of each session which has been guaranteed at the time of session admission. Although the average throughput of SD_MAC decreases with the increase in node speed, it still maintains the guaranteed throughput at higher ratio for all the admitted session in the network. Nevertheless, NFAC utilizes the opportunities proactively by trasmitting more violently to achieve better throughput. The effectiveness of SD_MAC approach provide opportunities for more nodes complete admitted sessions both in normal as well as in high mobility and traffic constrained environment. The combination of neural intelligence and fuzzy logic makes each node to self-organize and adapt to the dynamic changes making the network effective in decision making enhancing QoS in MANET.

5. Conclusions

Despite the rapid progress and the large volume of research activities made in the MANET, almost all research areas still impose many open issues. In this paper, we have proposed a novel "Software-Defined Multilayered Admission Control (SD_MAC) model" predominantly suitable for real-time mission critical applications. Primarily, the proposed scheme makes use of Neurofuzzy Inference-based Admission Control service engine which dynamically regulates real-time data transmission and accurately allocates resources in the face of network dynamics such as mobility and traffic overloads. Our protocol enhancement proves to make network more intelligent. Simulation results proves the proposed approach outperforms the existing approaches such as SWAN, StAC, DACME, and FuzzyQoS in terms of session admission ratio, session completion ratio, and low dropping probability, assuring QoS for end users. Our future work relies on challenges imposed in limitation on terrain models with interference posed due to environmental conditions.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[2] S. Chakrabarti and A. Mishra, "QoS issues in ad hoc wireless networks," *IEEE Communications Magazine*, vol. 39, no. 2, pp. 142–148, 2001.

[3] J. U. Duncombe, "Infrared navigation part I: an assessment of feasibility (periodical style)," *IEEE Transactions on Electron Devices*, vol. 11, p. 3439, 1959.

[4] S. Chen and N. Klara, "An overview of quality-of-service routing for the next generation high-speed networks: problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64–79, 1998.

[5] T. Bheemarjun Reddy, I. Karthigeyan, B. S. Manoj, and C. Siva Ram Murthy, "Quality of service provisioning in ad-hoc wireless networks: a survey of issues and solutions," vol. 4, no. 1, pp. 83–124, 2006.

[6] M. Karimi and D. Pan, "Challanges for quality of service in mobile ad hoc networks," in *Proceedings of the IEEE 10th*

Annual Wireless and Microwave Technology Conference, vol. 4, no. 2, pp. 129–153, Clearwater, FL, USA, April 2009.

[7] Y. Zhang and N. Ansari, "New Jersey institute of technology hiroshi tsoumada wireless telemedicine services over integrated IEEE 802.11/ WLAN and IEEE 802.16/ wi-max networks," *IEEE Wireless Communications*, vol. 10, pp. 1536–1284, 2010.

[8] C. R. Linn and J. S. Liu, "Qos routing in ad hoc wireless networks," *IEEE Journal on Selected Areas on Communications*, vol. 17, no. 8, pp. 1426–1438, 1998.

[9] L. Hanzo and R. Tafazolli, "The effects of shadow fading on QoS-aware routing and admission control protocols designed for multi-hop MANETs," *Wireless Communications and Mobile Computing*, vol. 11, no. 1, pp. 1–22, 2011.

[10] H. Luo, S. Lu, V. Bharghavan, J. Cheng, and G. Zhong, "A packet scheduling approach to QoS support in multi-hop wireless networks," *Mobile Networks and Applications*, vol. 9, no. 3, pp. 193–206, 2004.

[11] A. Bouchti, A. Haqiq, and S. Kafhali, "Analysis of quality of service performances of connection admission control mechanisms in OFDMA IEEE 802.16 network using BMAP queuing," *International Journal of Computer Science*, vol. 9, no. 1-2, pp. 302–310, 2012.

[12] Y. Dong and P. Du, "Cross-layer design of 2D queuing model for multi-hop wireless networks," *Wireless Personal Communications*, vol. 77, no. 3, pp. 1815–1832, 2014.

[13] V. Vidhyasanker, B. S. Manoj, and C. Siva Ram Murthy, "Slot allocation schemes for delay sensitive traffic support in asynchronous wireless mesh networks," *The International Journal of Computer and Telecommunications Networking*, vol. 50, no. 15, pp. 2595–2613, 2006.

[14] Y. Lin and V. W. S. Wong, "An admission control algorithm for multi-hop 802.11e based WLANs," *Computer Communication*, vol. 31, no. 14, pp. 3510–3520, 2008.

[15] L. Hanzo and R. Tafazolli, "Admission control schemes for 802.11-based multi-hop mobile ad hoc networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 78–108, 2009.

[16] G. Ash and D. McDysan, "Generic connection admission control (GCAC) algorithm specification for IP/MPLS networks," RFC 6601, Internet Engineering Task Force, Fremont, CA, USA, RFC 6601, 2012.

[17] C. Wang, W.-J. Yan, and H.-K. Lo, "Dynamic admission control and bandwidth reservation for IEEE 802.16e mobile Wi-MAX networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, 2012.

[18] G. H. Ahn, A. T. Campbell, A. Veres, and L. H. Sun, "SWAN: service differentiation in stateless wireless ad hoc networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 457–466, NewYork, NY, USA, June 2002.

[19] M. Asif, Z. Sun, and H. Cruickshank, "Admission control protocol in mobile ad-hoc networks provisioning QoS," in *Proceedings of the International Conference on Frontiers of Information Technology*, Abbottabad, Pakistan, December 2009.

[20] C. T. Calafate, M. P. Malumbres, J. Oliver, J. C. Cano, and P. Manzoni, "QoS support in MANETs: a modular architecture based on the IEEE 802.11e technology," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 5, pp. 678–692, 2009.

[21] B. Zhang and H. T. Mouftah, "QoS Routing for wireless ad hoc networks: problems, algorithms, and protocols," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 110–115, 2005.

[22] L. Khoukhi and S. Cherkaoui, "Intelligent QoS management for multimedia services support in wireless mobile ad hoc networks," *Computer Networks*, vol. 54, no. 10, pp. 1692–1706, 2010.

[23] A. Pandey, M. D. Nasir Ahmed, N. Kumar, and P. Gupta, "A hybrid routing scheme for mobile ad-hoc networks with mobile backbones," in *Lecture Notes in Computer Science*, vol. 4297, pp. 411–423, Springer, Berlin, Germany, 2006.

[24] A. Gnana Soundari and V. L. Jyothi, "Energy efficient machine learning technique for smart data collection in wireless sensor networks," *Circuits, Systems, and Signal Processing*, pp. 1–34, 2019.

[25] F. Cadger, K. Curran, J. Santos, and S. Moffet, "Opportunistic neighbour prediction using an artificial neural network," *International Journal of Advanced Pervasive and Ubiquitous Computing*, vol. 7, no. 2, pp. 38–50, 2015.

[26] V. Nair and E. G. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807–814, Haifa, Israel, June 2010.

[27] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, PMLR*, vol. 15, pp. 315–323, Ft. Lauderdale, FL, USA, 2011.

[28] L. Kleinrock, *Queueing Systems*, Vol. 2, Wiley, Hoboken, NJ, USA, 1975.

[29] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in intergrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[30] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted round-Robin cell multiplexing in a general purpose ATM switch chip," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1265–1279, 1991.

[31] N. Arora and S. Kaur, "Performance evaluation and fault repair mechanism for resilience of MPLS RSVPTE network with different application traffic load balancing across label switched path," *International Journal of Engineering Technology Science and Research*, vol. 2, no. 11, pp. 28–33, 2015.

[32] N. N. Kale and S. A. Waichol, "Performance analysis of MPLS network with traditional IP network in service provider environment," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 3, no. 4, pp. 1311–1316, 2014.

[33] A. R. Sulaiman and O. SalihAlhafidh, "Performance analysis of multimedia traffic over MPLS communication networks with traffic engineering," *International Journal of Computer Networks and Communications Security*, vol. 2, pp. 93–101, 2014.

[34] M. Kumar and S. Sangal, "Improving the usage of network resources using MPLS traffic engineering (TE)," *International Journal of Current Engineering and Technology*, vol. 5, no. 1, pp. 261–265, 2015.

[35] E. Kafhaliabdelali and A. KrimHaqiq, "Performance analysis for bandwidth allocation in IEEE 802.16 broadband wireless networks using BMAP queuing," *International Journal of Wireless & Mobile Networks*, vol. 4, no. 1, pp. 139–154, 2012.