Journal of Research in Science and Engineering (JRSE)
ISSN: 1656-1996 Volume-8, Issue-4, January 2026
DOI: 10.53469/jrse.2026.08(04).14

Dynamic Routing in Software-Defined Networks Using Multi-Agent Deep Deterministic Policy Gradient

Ohm Patel[1], Abdul Baes[2]

[1]Assistant Professor, Department of Computer Science, Bishop Ambrose College, Coimbatore -641 035, Tamil Nadu, India
ohm80@gmail.com

[2]Assistant Professor, Department of Computer Science, PPG college of arts and science, Coimbatore -641 035, Tamil Nadu, India
abdulbaes@ppg.edu.in

Abstract: ***The separation of the control plane (CP) from the data plane (DP) facilitates in Software Defined Networking (SDN) centralizing the network management. SDN is an innovative network design. This separation will help in the configuration of the network, management of the network and optimization of the network. This separation also contributes more programmable and adaptable control over the network activities. Here, adjusting to high-dimensional (HD) state-action (S-A) spaces and quickly shifting network conditions are not contributed by the conventional Routing Protocols (RP). For the purpose of overcoming those limitations in SDN network routing, the (MA) Multi-Agent (DDPG) Deep Deterministic Policy Gradient (MADDPG) algorithm was suggested in this study. Decentralized agents were enabled by this MADDPG, and network nodes are stimulated by these agents for learning the Optimal Routing Policies (ORP) collaboratively. During training, the global network state is considered. Then, a scalable, and adaptive Decision Making (DM) was facilitated by this approach, as it integrates centralized training and decentralized execution. Important objectives are optimized by this model, as it attains minimizing latency, balancing network loads, and maximizing throughput. Adapting to dynamic traffic patterns and faults were facilitated by the MADDPG-based routing with the Reinforcement Learning (RL). This integration will also support in ensuring the robust and Real-Time (RT) operation. Simulation was conducted, and the outcomes of the simulation indicates that the suggested MADDPG performs better than the current RP by delay, Packet Loss (PL), and throughput (T). MADDPG became an effective method for future SDN settings.***

Keywords: Software Defined Networking (SDN), Network routing, Optimal Routing Policies (ORP), Multi-Agent Deep Deterministic Policy Gradient (MADDPG) procedure

1. Introduction

The separation of the CP from the DP facilitates in SDN centralizing the network management. SDN is an innovative network design [1]. This separation also contributes more programmable and adaptable control over the network activities. Here, the network operators can use the technology in modifying the network behavior dynamically, and it was facilitated by the decoupling of DP and CP. This separation also enhances the ability in responding to several needs and settings. Global perspective of the network was provided by the centralized ability of SDN. Then, network resource optimization, and overall performance was optimized by the centralized nature of the SDN. The Traffic Engineering (TE) and Network virtualization have advanced significantly by this architectural shift. For the purpose of managing large-scale networks, a robust framework was offered by this architectural shift. Conventional RP in SDN has several difficulties despite its benefits. Dynamic and complex nature of modern networks are not effectively handled by the conventional RP in SDN. The static nature of these algorithms thus limits their performance in backgrounds with rapidly changing traffic patterns and network topologies. Conventional methods often face difficulties in attaining key performance objectives like minimizing delay, balancing network loads, and maximizing T. These conventional methods have the following drawbacks, such as resilience of the network to failures and ability of the conventional methods in make RT decisions. Responding to dynamic and heterogeneous network backgrounds are not facilitated by the conventional algorithms, because these conventional methods often depend on static configurations and pre-defined rules. This will lead to inefficient resource utilization, degraded quality of service (QoS), and bottlenecks result from the inability of these conventional methods in swiftly adapting to dynamic networks. In case of unpredictable traffic patterns, adapting to the increasing intricacy and size of modern networks are not managed by these algorithms because these algorithms are static in nature.

Then, the incorporation of RL in SDN routing helps in overcoming those limitations, RL is a type of Machine Learning (ML). So, this integration has gained attention nowadays [2]. The adaptability and smarter routing decisions are then enhanced by the RL methods like MADDPG algorithm. These MADDPG algorithm have the potential for enhancing adaptability and smarter routing decisions. The metwork's current state is considered by the MADDPG, and this consideration enables the MA in learning and DM. Finally, a more responsive and efficient routing mechanism was also offered by this MADDPG method.

The Deep RL (DRL) are presented for revolutionizing TE in SDN. The advantages of Deep (NN) Neural Networks (DNN) and conventional RL are combined. Thus, more effective handling of complex network situations is also facilitated by this integration. When comparing the suggested DRL-based routing solutions, like the DDPG with other conventional methods, the suggested method effectively manages the dynamic traffic patterns and HD state-action spaces, and it was stated by Kim et al. (2022) [3].

In this study, the application of MADDPG in SDN backgrounds are presented for the purpose of resolving those

issues. This suggested method may offer a more adaptive and novel technique, and these novel methods will aim to improve the routing efficiency and network performance [4]. This method utilizes the decentralized nature of MADDPG, a collaborative learning and performing optimal RP in RT was then facilitated by this decentralized nature of MADDPG [5,6]. This suggested method is capable of handling dynamic traffic patterns effectively. It also improves the resource utilization, and resilience of the network. It is clear that the crucial metrics like latency, T, and PL were improved. Thus, a more robust and effective SDN background was attained.

The remaining study is systematized as follows: Section 2 examines some of the most recent methods of the SDN RP. For effective routing, the suggested MADDPG method and SDN systems are thoroughly explained in Section 3. The result and discussion are presented in section 4. The conclusion of the study is summarized in section 5.

2. Literature Review

A novel MADDPG integrated MA framework in SDN was introduced by Dake et al. [2021] [7]. These frameworks are used for the purpose of detecting and preventing malicious DDoS traffic in the network and optimizing Multipath Routing (MR). Here, 2 agents of the MARL are collaborated in a similar background for completing the network optimization task faster. To mathematically represent the State (S), Action (A), and Reward (R) of the suggested method, the Markov Decision Process (MDP) was utilized. In these models, the MADDPG algorithm was then updated. The following network metrics like intrusion detection (ID), jitter, PL Rate (PLR), delay, and bandwidth (BW) utilisation were employed for assessing the effectiveness of the methods. For assessing purpose, the suggested MADDPG-based framework was contrasted with DDPG. From the outcomes of the analysis, significantly improved network metrics was attained by the 2 agents.

The Decentralised Execution with DE-MADDPG algorithm framework was adopted by Ke et al. [2024] [8]. According to local network data, the protocols and protocols parameters are directly adjusted by the unmanned aerial vehicles (UAVs), and it was facilitated by this framework. The transmission delay of control signals is not considered for optimizing the network structure and overall performance. Then, the ns3-gym simulation platform was employed for conducting the performance evaluation. Comparing the suggested method to other methods like MADDPG and deep Q-network (DQN) for evaluation.

A MA RL framework in SDN-IoT (Internet of Things) was suggested by Dake et al [2021] [9]. This framework helps in effectively detects and mitigate DDoS attacks and managing rapid traffic events without compromising benign traffic. In the framework, a MADDPG algorithm was built, then researchers used Mininet to simulate a 200-node topology with increased BW and transmission rate. The simulation was conducted, and from the outcomes of simulation, it is clear that the recommended system executes superior when compared to the DDPG algorithm by the following network parameters: PL, ID, jitter, latency, and BW utilisation of network flows.

The MADDPG-based Traffic Control and Multi-Channel Reassignment (TCCA-MADDPG) algorithm was suggested by Wu et al. [2020] [10]. This framework was suggested for optimizing the objection function (OF) , accomplishing traffic control and channel reassignment. As a part of channel state data, the dynamic and complexity of core backbone networks are managed by the traffic prediction. For effectively utilize the time continuity of the channel S, the LSTM layer is added to the NN (Neural Network) in this study. Form the outcomes of the simulation, the suggested method performs superior when compared to the current approaches.

For multi-path TCP (MPTCP) in heterogeneous wireless networks, an Energy-Efficient (EE) scheduling strategy Was suggested by Arain et al. [2023] [11]. Minimum delay, high T, and less Energy Consumption (EC) was ensured by this approach. An agent may cooperate with other agents, and agents can control every sub-flow of the MPTCP by using the MADDPG algorithm. The decentralized policies are learnt by this method through the centralized training and decentralized execution. Then, the scheduling problem is modelled using a MA DM tasks. When comparing the suggested EE scheduling scheme with current scheduling methods, notable energy conservation and preserving lower delay and higher T was demonstrated by the suggested method.

3. Proposed Methodology

The suggested methodology leverages the MADDPG algorithm for efficient SDN routing. In this approach, decentralized agents represent network nodes and collaboratively learn optimal routing policies through reinforcement learning. Centralized training ensures that the agents are aware of the global network state, while decentralized execution enables scalable and adaptive decision-making. Optimising important network goals like maximising T, balancing traffic loads, and reducing latency is the main focus of the MADDPG framework. This algorithm may be adjusted to dynamic traffic patterns and network failures, and it was facilitated by the integration of RL. It ensures robust and efficient operation. Figure 1 shows the general flow of the suggested approach.

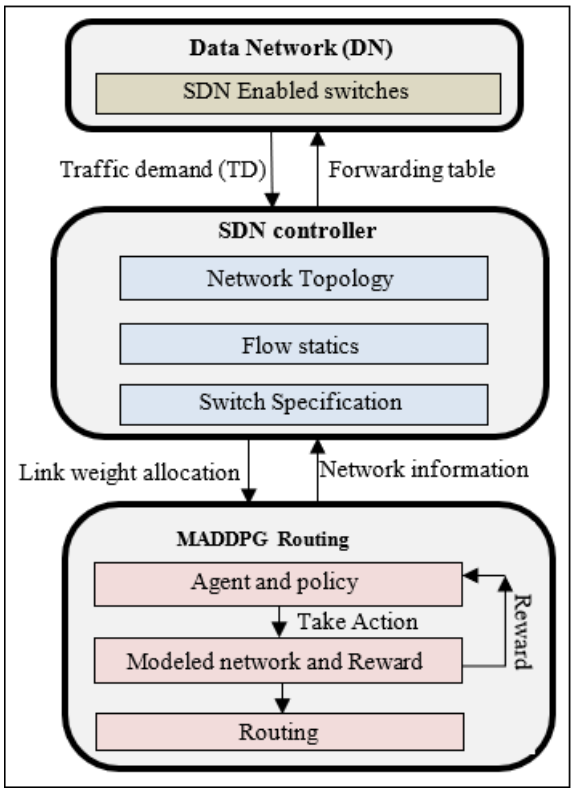

Figure 1: Overall Process of Proposed System

3.1 System Model

Examine a backbone network that consists of switches with SDN capabilities. Many servers or clients are among the traffic sources connected to the edge switches. [12]. In this case, every edge switch is assumed to represent the backbone network's source or destination. The SDN-based network contains N switches. The expression $V = [v_1, \dots, v_N]^T$ represents each switch. μ_n represents the nth switch's service rate. The traffic routing within a single autonomous system is the main topic. The expression for a communication network G is $G = (V, E)$. Then, the set of links e between the switches are denoted as E.

If v_i, v_j, are any 2 switches $V\ (i \neq j)$,it assumes that there exists at least one path forwarding (PF) $p_{i,j} = \{e_1, e_2, \dots, e_{|p|}\}$ that can send the traffic from $v_i$to v_j. The formula $p^*_{i,j} = \{e_1, e_2, \dots, e_{|p^*|}\}$ indicates the Shortest Path (SP) among the paths. In this case, a weighted SP algorithm on G is used to determine the SP. Here $w_m(0 \leq m \leq |E|)$ represents the weight value for link e_m. as $p^*_{i,j} = \{v_1^{i,j}, v_2^{i,j}, \dots, v_{|p|}^{i,j}\}$ is another method to describe this path, when transmitted across the SP, $v_l^{i,j}$ denotes the lth switch. Then, $v_i\ and\ v_j$ are equivalent to $v_1^{i,j}$and $v_{|p^*|}^{i,j}$. Since $M_t$is the flow count in the network at time t, let $f_t^k\ (0 \leq \leq M_t)$ represent the kth traffic flow at t.

In this context, a flow is a pair of source and destination. As the flow passes via switches along the route, there are delays in the Data transmission (DT)and processing of data.
Based on the estimated SP, f_t^k is routed through specific switches in the network routing system.

Assume that the Poisson arrivals have exponentially distributed service times and a rate of λ_t^k. Here, all switch has a restricted system dimension, and the data processing and DT delay can be represented as an M/M/1/K queue. Next, the following can be used to determine the anticipated delay in v_n at t equation 1,

$$E[d_n(t)] = \frac{E[N_n(t)]}{\lambda_n(t)\left(1-P_b^n(t)\right)} \quad (1)$$

Here, the total arrival rate into v_n at t is denoted by $\lambda_n(t)$.Due to a buffer overflow in v_n, a packet is lost at t and its probability is denoted as $P_b^n(t)$. The switch's packet count at t is represented by the queue occupation, $N_n(t)$. In this case, $P_b^n(t)$can be found in this way at below equation 2,

$$P_b^n(t) = \frac{(1-\rho_n(t))(\rho_n(t))^{K_n}}{(1-\rho_n(t))^{K_n+1}} \quad (2)$$

Here, $\rho_n(t) = \lambda_n(t)/\mu_n$. The traffic intensity at switch n and the switch's overall potential of the model are indicated by K_n. Additionally, the following illustrates a way to determine the predicted queue task in equation 3,

$$E[N_n(t)] = \begin{cases} \frac{\rho_n(t)}{1-\rho_n(t)} - \frac{(K_n+1)(\rho_n(t))^{K_n+1}}{1-(\rho_n(t))^{K_n+1}} & if\ \rho_n(t) < 1 \\ \frac{K_n}{2} & if\ \rho_n(t) = 1 \end{cases} \quad (3)$$

The following equation 4 is the E2ED of f_t^k sent over $p^*_{i,j}$, with a consideration that the link's propagation delay is extremely small [15].

$$D_{e2e}^k(t) = \sum_{n \in path(p^*_{i,j})} E[d_n(t)] \quad (4)$$

Here, $path(p^*_{i,j})$ represents the collection of switches that create the forwarding path from v_i to v_j. The average E2ED of the network's flows may then be determined by applying the following equation 5,

$$D_{e2e}^{avg}(t) = \frac{1}{M_t}\sum_{k=1}^{M_t} D_{e2e}^k(t) \quad (5)$$

Additionally, the following techniques can be used to ascertain the expected network loss traffic at t as well as the expected loss traffic of switch n in equation 6 and equation 7,

$$E[L_n(t)] = \lambda_n(t) P_b^n(t) \quad (6)$$

$$E[L_{tot}(t)] = \sum_{k=1}^{N} \lambda_n(t) P_b^n(t) \quad (7)$$

Without sacrificing the data network's performance, the network modelling approach helps the learning agent create a neural model that is well-trained. The method can be easily expanded to multi-path routing (MPR), but current work assumes Single-Path Routing (SPR), like OSPF. According to the RP, the DN's S and R can change for the same set of Link Weights (LW). When the training is properly redone, the RL agent can adapt to various surroundings since the RP is seen as a component of the background under the RL framework.

3.2 MDP Framework for LW Allocation

From one switch to the next in line, the Aggregated Traffic Volume Matrix (ATVM) shows the traffic rate. With $t_{i,j}$ being the quantity of traffic from the ith switch to the jth switch, let $T = \left[t_{i,j}\right]_{N,N}$ represent the ATVM of the DN. The TD of data flows and their network routing patterns determine the network's ATVM for a certain network design. Because it depicts both the network's link utilisation and the connectivity architecture among its switches, the agent can comprehend the dependency among the links by this representation of network S. The DRL agent and the SDN controller fail to interact during the training phase in the suggested modeling-based approach [13]. This indicates that during the learning phase, the agent fails to evaluate the total traffic volume at switches. The DN model and routing path decisions are exploited to compute ATVM. A RP and the LW from the previous A in determine ATVM. The value determined for the deployed link weight allocation (WA) and TD in the fine modeling-based learning environment in the suggested technique is identical to the aggregated traffic volume value that each switch in the actual network observes [13]. The DRL agent can use this consistency to train a neural model that can be applied to the DN. Then, without affecting network performance, the agent was allowed to implement unconstrained routing policies. These significant S-R pairs are also used during the experience replay process and are kept in the off-policy DRL agent's replay buffer.

S and A SPACE:

The observation for every switch v_n at t can be expressed as $o^n(t)$ using equation 8.

$$o_t^n = [t, K_n, N_n(t), \lambda_n(t), L_n(t), \rho_n(t), d_n(t)] \quad (8)$$

Here, timestep is denoted as t. The switch's index is denoted by n. N is the number of switches, and k is the flow index. The flows count is denoted as M_k. The nth switch is v_n. The nth switch's service rate is denoted by μ_n. The kth traffic flow at t is denoted by f_t^k. The PL probability in v_n at t is $P_b^n(t)$. The overall system capacity of $v_n$is denoted by K_n. The Queue occupation at t is $N_n(t)$. The total arrival rate into v_n at t is denoted by $\lambda_n(t)$. The expected loss traffic v_n at t is denoted by$L_n(t)$. The use of v_n at t is denoted by $\rho_n(t)$. The expected delay in v_n at t is denoted by $d_n(t)$. In the suggested approach, these observations are derived from the agent's modelled network rather than measurements on every switch in the DP.

The SDN controller reports network information, including switch specifications and network topology, which is used to build the modelled network.The SP of flows is computed in the modelled network at every t based on the LW assigned by the previous A. The network S at t, is represented as S_t. Lastly, s_t can be found as follows equation 9 and equation 10,

$$S_t = \{s_t^{i,j} | i, j \in V\} \quad (9)$$

$$s_t^{i,j} = \min\left(1, \frac{1}{\mu_{max}} \sum_{k=1}^{M_t} \lambda_t^{k(i)} x_{ij}^k(t)\right) \quad (10)$$

Here, $\lambda_t^{k(i)} = \prod_{l \in path(src(k) \to i)} \left(1 - P_b^l(t)\right)$. λ_t^k and $x_{ij}^k(t)$ provide the binary indicator that shows whether the connection from v_i to v_j is a component of the way that the kth flow is transmitted according to the SP.

The switches with maximum service rate are denoted by μ_{max}, and the component range of state is adjusted to [0, 1] using the $min(.)$function. Consider that the component $t_{i,j}$ of ATVM normalised by μ_{max}corresponds to the state $s_t^{i,j}$. In the meantime, the DRL agent decides how to distribute the LW based on the s_t at each t. The following is the definition of the action space on below equation 11,

$$a_t = a_t^1 \times a_t^2 \times \ldots \times a_t^{|E|} \quad (11)$$

Here, the weight value given to the mth link at t is indicated by $a_t^m (0 \leq m \leq |E|)$. Every weight has a range defined as $a_t^m \in [w_{min}, w_{max}]$. The LW minimum and maximum values are represented by the symbols $w_{min}$and w_{max}. Since the weighted SP method determines packet forwarding, the DRL agent's action shows the LW, and LW allocation indicates the network's routing policy. Figure 2 provides a simple illustration of how to use a modeling-based environment to compute the S and the associated delay and loss deprived of having an impact on the data network. There are 2 TD on the network, which consists of six switches.

The SDN controller delivers DN to the learning agent and controls packet forwarding in accordance with its routing policy [14]. Without interfering with the DN, the DRL agent trains the NN for a definite topology of the network. The modelled network is used to compute the DRL data, including the network state, average E2ED, and expected total PL.

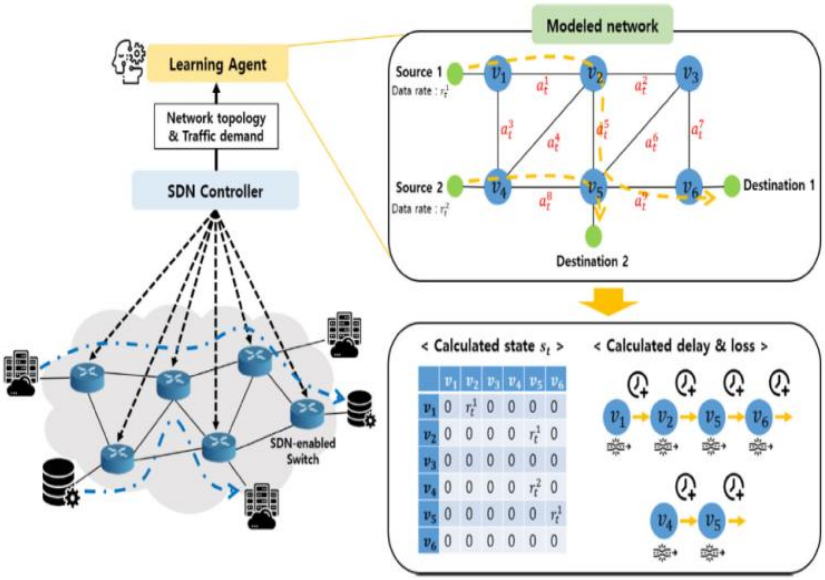

Figure 2: Basic Instance of Modeling-Based Procedure

$\mathbb{P}_{sa}: (s, a) \rightarrow s_{t+1}$ is a representation of the probability, when action a_t is performed, the network RP will change from state s_t to state s_{t+1}. The probability of choosing an $a_t, and\ s_t$ given by policy π can be used to calculate the transition probability $\mathbb{P}_{sa}$ then the WA to every link is adjusted continuously. The suggested RP aims to accomplish two primary objectives: (2) reducing the rise in PL at switches, and (3) lowering the average E2ED of flows. The PL reward $r_p(t)$ and delay performance reward $r_d(t)$ should be defined as follows at time step t of equation 12 and equation 13,

$$r_d(\mathrm{t}) = 1 - \frac{D_{e2e}^{avg}(t)}{\sum_{n \in path(p_{max})} \frac{Kn}{\mu_n}} \quad (12)$$

$$r_p(\mathrm{t}) = 1 - \frac{L_{tot}(t)}{\sum_{n=1}^{N} \lambda_n(t)} \quad (13)$$

The path with the greatest number of hops is presumed to be p_{max} in this case. Let R represent the R function that yields a value showing whether LW are distributed in a way that minimises the average E2ED via the routing method while accounting for PL reductions caused by bottlenecks. The R is defined as follows when a_t is performed in s_t:

$$R(s_t, a_t) = \alpha r_d(t) + (1 - \alpha) r_p(t) \quad (14)$$

In order to balance the weights of network packet loss (PL) and delay in equation 14, α is utilized as the weight factor and $R(s_t, a_t)$ has a range of [0, 1]. R seeks to minimise a weighted linear combination of PL and network delay performance metrics, as indicated in (14). Different definitions of the R, such as maximising T or minimising maximum link utilisation, can be applied based on the requirements of network operation. Define the total expected discounted R under policyπ as follows, it considers the effect of present action on future R:

$$R_t^{\pi} = R(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i . R(s_{t+i}, a_{t+i}) \quad (15)$$

Here, equation 15 is the relevance of future R is determined by the discount factor $\gamma \in [0, 1]$ from t+ 1 to the infinite t. The total of the present R at t and the discounted R from t+ 1 to the infinite t is known as R_t^{π}.

3.3 MADDPG Algorithm for Network Routing Process

The MADDPG algorithm applied to SDN optimizes routing decisions across multiple agents, each representing an SDN controller. Each agent learns a deterministic policy in a MA environment using MADDPG, which is an extension of the DPG method [15]. The algorithm uses decentralised execution and centralised training.

All agents have access to the global S data during training, but they make their decisions on their local observations when they are really in A. The network model used in this study consists of multiple SDN controllers managing different sections of the network. Each controller is treated as an agent within the MADDPG framework.

The training process involves simulating various network scenarios to expose the agents to a wide range of conditions. In this SDN-based environment, the SDN controller handles the global network state and provides real-time updates to the agents, which are individual network devices (e.g., switches, routers).Agents are network elements that make decisions on routing, traffic management, or other network parameters based on local observations and feedback from the SDN controller, and it optimizes the total network performance by adjusting configurations and directing the actions of agents using a global view. In the figure 3 described the MADDPG model, here O represents the observations made by the agents, including network state information like traffic and node load. a and a^m represent the actions taken by the agents during execution and training phases, respectively, for routing and policy optimization.

Exploration and Initialization: Each agent's actor network (AN)and critic networks are initialised to start the learning process. The agent's action at t is elected by the AN, while the critic evaluates the action taken. The exploration strategy is based on the noise process to ensure sufficient exploration during the training phase. At every t, the agent chooses A at

based on its current policy μ(st) (from the AN), along with an exploration noise Nt as equation 16,

$$a_t = \mu(s_t) + N_t \quad (16)$$

Given the s_t, the AN predicts an action denoted by $\mu(s_t)$. N_t represents exploration noise, which encourages the agent to explore different actions.

Critic Network Update: The loss function (LF) is minimised in order to update the critic network. This critic network contrasts the target value yi with the Q-function's predicted value. The R obtained at time step i and the future Q-value predicted by the target critic network are used to calculate this target value in below equation 17 and equation 18.The L is minimised in order to update the critic network:

$$L(\theta Q) = \frac{1}{H}\sum_{i=1}^{H}(y_i - Q(Si, ai|\theta Q))^2 \quad (17)$$

where:

$$y_i = \gamma_i + \Upsilon Q'(s_i + 1, \mu'(s_i + 1|\theta'_\mu)|\theta'_Q \quad (18)$$

Where, $Q'(s_i + 1, \mu'(s_i + 1|\theta'_\mu)|\theta'_Q$are the target critic and actor networks, respectively and ϒ is the discount factor, and H is the number of experiences in the replay buffer.

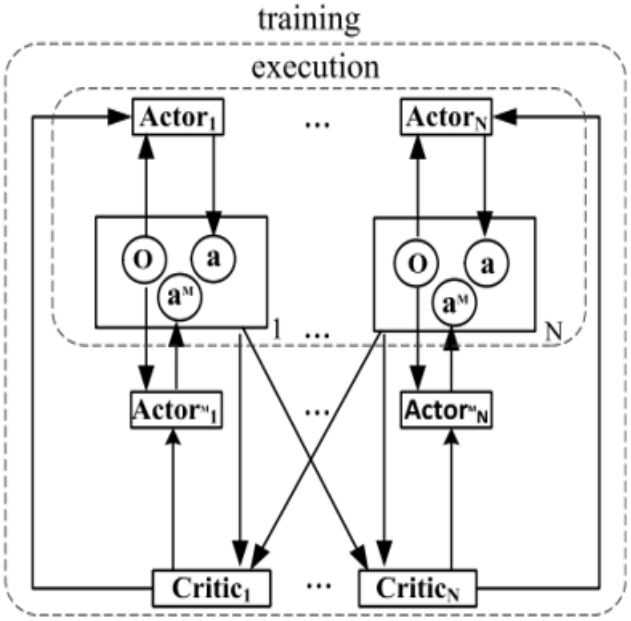

Figure 3: MADDPG System Flow

Actor Network Update: The Policy Gradient (PG) approach is used for updating the AN. The objective of PG in relation to the AN parameter is calculated. The expected R can be maximized and also facilitates the agent in making actions.The PG is used to update the actor network as equation 19,

$$\nabla\theta\mu^J(\mu) \approx \frac{1}{H}\sum_{i=1}^{H}\nabla_a Q\,(S, a|\theta_a)\nabla_{\theta\mu}(s_i|\theta\mu) \quad (19)$$

The gradient of the AN with respect to the A is $\nabla_a Q(S, a|\theta_a)$. The gradient of the critic network with respect to the A is $\nabla_{\theta\mu}(s_i|\theta\mu)$. The AN's OF is $j(\mu)$, and the AN's parameters are θμ.

Target Network (TN) Update: Both the TN of actor and critic are updated using a soft update technique. This technique gradually shifts the TN' weights towards the weights of the existing networks in order to guarantee stable learning. The following updates are made to the TN:

$$\theta'_Q \Leftarrow T^\theta\mu + (1 - T)\theta'_Q \quad (20)$$

$$\theta'_\mu \Leftarrow T^\theta\mu + (1 - T)\theta'_\mu \quad (21)$$

Where, equation 20 and equation 21 is the current critic and AN parameter are denoted by θ_Q and θμ, the parameters of the AN are θ'_μ and the target critic is θ'_Q, T is the soft update parameter, typically set between 0 and 1, controlling the rate of updating.

Reward Function: Both network performance and resource consumption, including BW utilisation, latency, congestion, and EC (if applicable), must be considered by the reward function. Actions that improve network performance, lessen congestion, and raise the general QoS should be rewarded by the reward function. At t, the reward for every agent (network element) is specified as follows equation 22,

$$R(s_t, a_t) = -(\alpha \cdot Congestion(s_t) + \beta \cdot Latency(s_t) + \gamma \cdot Bandwidth\_Efficiency(a_t)) \quad (22)$$

Where, $Congestion\,(s_t)$: A penalty term for network congestion at state s_t, $Latency\ s_t$: A penalty term for network latency at state s_t, $Bandwidth\_Efficiency(a_t)$: A term to encourage efficient use of available bandwidth given the action a_t, α, β, γ: Weights that balance the importance of congestion, latency, and bandwidth efficiency in the reward function.

Centralized Control (SDN) and Distributed Execution: Global optimisations like as load balancing, resource allocation (RA), and routing decisions are made possible by the SDN controller. This SDN controller offers a centralised outlook of the overall network.

Following decentralised execution, the agents (network elements) use the SDN controller's global insights to make decisions on their own, based on their local states.

Through this integration, the network's overall performance can be optimised while dynamically adapting to shifting traffic conditions. Every agent receives feedback from the centralised controller while using its local state data, such as traffic load and BW availability. It enhanced RT's decisions about network management.

MADDPG Algorithm for SDN-based Routing Optimization

Step 1: For every agent, Set the AN and critic networks
Step 2: Given the same weights as the original networks, Set the AN and critic networks
Step 3: Set the replay buffer DD to store experiences.
Step 4: Set the update rate beta for the TN.

Step 5: Repeat for each episode from 1 to N:
Step 6: For each agent pp:
Step 7: Select an action using the AN and add exploration noise to encourage exploration.
Step 8: Execute the selected action, observe the reward rr, and the next state s′s'.
Step 9: Store the transition (s,a,r,s′)(s, a, r, s') in the replay buffer DD.
Step 10: Update the current state ss to the next state s′s'.
Step 11: End the loop for each agent.
Step 12: If the episode is a multiple of the update interval:
Step 13: For each agent pp:
Step 14: Sample a random minibatch of experiences from the replay buffer DD.
Step 15: Compute the target value yiy_i using the target critic and AN: equation 18
Step 16: Update the critic network by minimizing the loss function: equation 17
Step 17: Update the AN using the PG: equation 19
Step 18: Perform a soft update of the TN for stability: equation 20 and 21
Step 19: End the loop for each agent.
Step 20: End the episode loop.

By combining centralized training with decentralized execution, the framework ensures scalability and robustness against network failures.

For dynamic routing in SDN, a significant advancement was attained by the application of MADDPG in comparison to conventional methods [16]. The efficiency, scalability, and resilience of SDN networks are enhanced by the suggested method, as it enables decentralized agents to learn collaboratively and adapt to changing the conditions of the network. Here, minimizing latency, reducing PL, and maximizing T, was attained by the suggested method. This suggested method ensures efficient network operation. Thus, scalability and robustness against network failures was ensured by integrating the centralized training with decentralized execution

4. Result and Discussion

In SDN-based network, the performance comparison of suggested RP and other protocols was evaluated by the simulation outcomes, and it was presented in this section. In the simulations, stable-baseline framework for the MADDPG method and MATLAB R2021b for network topology construction was utilized. The simulations were done on a server equipped with an Intel i9-10940X CPU running at 3.3 GHz, an NVIDIA Quadro RTX 6000 with 24 GB of memory, and CUDA 11.1 in order to provide the numerical results. For evaluation purpose, the following metrics are used.

The total of the delays experienced at a series of intermediate nodes on the way to the destination is the E2ED of each packet. Each delay consists of a variable component, such as processing and queueing delays at the nodes, and a fixed component, such as transmission and propagation delays.

The Packet Delivery Ratio (PDR) can be expressed as following as equation 23,

$$R_{Success} = \frac{P_{succ}}{P_{all}} \times 100\% \quad (23)$$

where $R_{Success}$represents the ratio of packet successful, P_{succ}denotes the total packets that successfully sent by the router, P_{all} indicates the total packets that are transmitted through the router.

The packet loss ratio can be expressed as follows equation 24,

$$R_{drop} = \frac{P_{drop}}{P_{all}} \times 100\% \quad (24)$$

where R_{drop}represents the ratio of packet loss, P_{drop}denotes the number of packets dropped by the router, P_{all} indicates the total number of packets transmitted through the router. T is the amount of data that transfers across a network in a given period of time. Typically, it is expressed in megabits per second (Mbps) or bits per second (bps).

These metrics collectively demonstrate the proposed algorithm's superiority in ensuring reliable, efficient, and scalable data transmission within SDN-based networks.

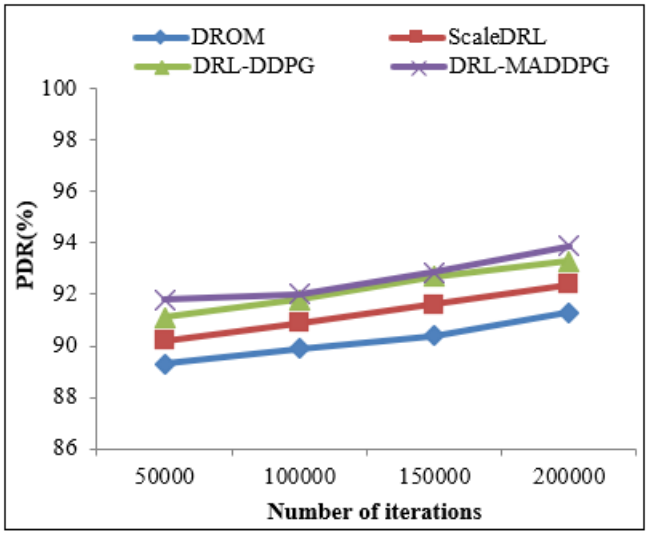

Figure 4: PDR VS. Routing Protocols

Figure 4 shows that the proposed DRL-MADDPG system achieves the highest Packet Delivery Ratio (PDR) of 94.20% at 200,000 iterations. Other methods, including DROM, ScaleDRL, and DRL-DDPG, exhibit lower PDR values of 91.30%, 92.40%, and 93.30%, respectively. Thus, effective PDR was ensured by the efficacy of the suggested method.

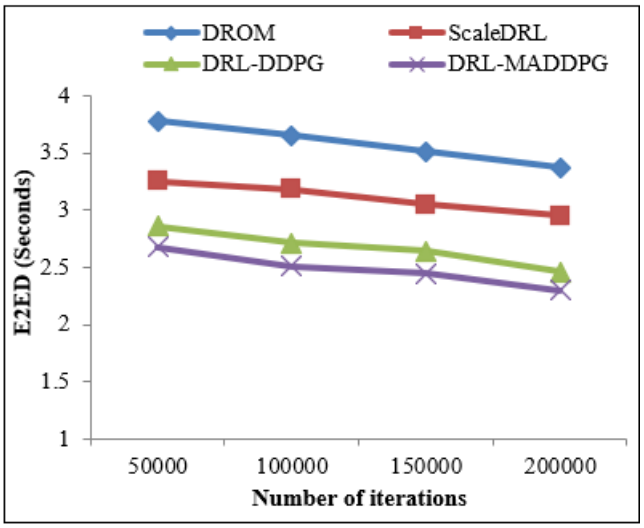

Figure 5: E2ED VS. Routing Protocols

The suggested DRL-MADDPG system attains minimal E2ED of 2.2 seconds at 200,000 iterations, and it was presented in Figure 5. But, other methods like DROM, ScaleDRL, and DRL-DDPG attains highest delay of 3.4 seconds, 2.8 seconds, and 2.5 seconds. The communication delay was effectively reduced by the suggested method, and it was demonstrated.

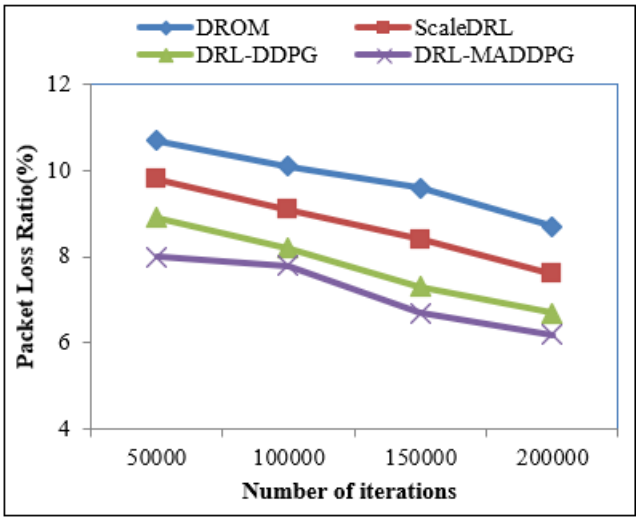

Figure 6: PLR VS. Routing Protocols

Figure 6 presents the PLR VS. RP over several iterations. Lowest PLR of about 5% was attained by the suggested method at 200,000 iterations. Then 6% PLR was attained by the DRL-DDPG, and it holds 2nd place. Then, ScaleDRL and DROM have greater PLRs of about 7% and 9%. During communication, the PLR was minimized by the DRL-MADDPG system minimises PLR with exceptional reliability, and it was highlighted in the outcomes.

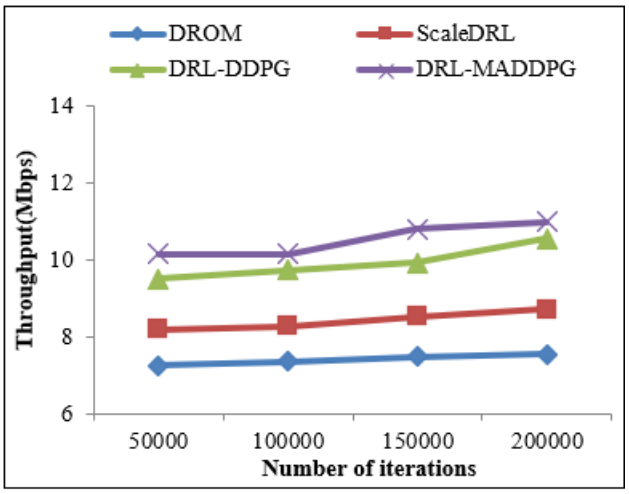

Figure 7: Throughput VS. Routing Protocols

Maximum T of about 12.5 Mbps at 200,000 iterations was attained by the suggested DRL-MADDPG system. These values indicate that the suggested method performs superior in contrast to the conventional methods, and it was presented in Figure 7. DRAM and ScaleDRL attain lower T of about 8 Mbps and 9.5 Mbps, and DRL-DDPG comes in second with a T of about 11 Mbps. These outcomes show how well DRL-MADDPG performs when it comes to network throughput optimisation.

5. Conclusion

A novel technique was introduced for dynamic routing in SDN with the MADDPG algorithm. The centralised training and decentralised execution was employed by the system. By this application, the system enables agents to collectively optimise RP and dynamically adapt to network changes. The simulations are conducted by comparing the suggested RP to another RP. Then, the outcomes of the simulation indicate that the suggested method executes better in maximising T, decreasing PL, and minimising E2ED. A scalable, reliable, and effective network functioning was ensured by the suggested method even in the face of constantly shifting traffic patterns and problems. Then, it combines its decentralised DM capability with global network awareness during training. Tus, the adaptability and reliability was improved. Finally, future work emphasis on incorporating the robust security mechanisms for handling new risks like hostile attacks and DDoS attacks. An effective way for creating smart, adaptable, and secure SDN solutions was offered by this study for satisfying the demands of next-generation networks.

References

[1] D. M. Casas-Velasco, O. M. C. Rendon, N. L. da Fonseca, “Intelligent routing based on reinforcement learning for software-defined networking,” IEEE Transactions on Network and Service Management, 18(1), pp. 870-881, 2020.

[2] A. Al Hayajneh, M. Z. A. Bhuiyan, I. McAndrew, “Improving internet of things (IoT) security with software-defined networking (SDN),” Computers, 9(1), p. 8, 2020.

[3] G. Kim, Y. Kim, H. Lim, "Deep reinforcement learning-based routing on software-defined networks," IEEE Access, 10, pp. 18121-18133, 2022.
[4] G. Wu, H. Wang, H. Zhang, Y. Zhao, S. Yu, S. Shen, "Computation offloading method using stochastic games for software-defined-network-based multiagent mobile edge computing," IEEE Internet of Things Journal, 10(20), pp. 17620-17634, 2023.
[5] A. M. Seid, A. Erbad, "Multi-agent RL for SDN-based resource allocation in HAPS-assisted IoV networks," In ICC 2023-IEEE International Conference on Communications, pp. 1664-1669, 2023.
[6] K. Boutiba, M. Bagaa, A. Ksentini, "Multi-Agent Deep Reinforcement Learning to enable dynamic TDD in a multi-cell environment," IEEE Transactions on Mobile Computing, 2023.
[7] D. K. Dake, J. D. Gadze, G. S. Klogo, H. Nunoo-Mensah, "Multi-agent reinforcement learning framework in sdn-iot for transient load detection and prevention," Technologies, 9(3), p. 44, 2021.
[8] Y. Ke, K. Huang, X. Qiu, B. Song, L. Xu, J. Yin, Y. Yang, "Distributed Routing Optimization Algorithm for FANET Based on Multi-agent Reinforcement Learning," IEEE Sensors Journal, 2024.
[9] D. K. Dake, J. D. Gadze, G. S. Klogo, "DDoS and flash event detection in higher bandwidth SDN-IoT using multiagent reinforcement learning," In 2021 International Conference on Computing, Computational Modelling and Applications (ICCMA), pp. 16-20, 2021.
[10] T. Wu, P. Zhou, B. Wang, A. Li, X. Tang, Z. Xu, K. Chen, X. Ding, "Joint traffic control and multi-channel reassignment for core backbone network in SDN-IoT: a multi-agent deep reinforcement learning approach," IEEE Transactions on Network Science and Engineering, 8(1), pp. 231-245, 2020.
[11] Z. A. Arain, X. Qiu, C. Xu, M. Wang, M. Abdul Rahim, "Energy-Aware MPTCP Scheduling in Heterogeneous Wireless Networks Using Multi-Agent Deep Reinforcement Learning Techniques," Electronics, 12(21), pp. 4496, 2023.
[12] G. Kirubasri, S. Sankar, D. Pandey, B. K. Pandey, V. K. Nassa, P. Dadheech, "Software-defined networking-based Ad hoc networks routing protocols," In Software defined networking for Ad Hoc networks, Cham: Springer International Publishing, pp. 95-123, 2022.
[13] Y. J. Wu, P. C. Hwang, W. S. Hwang, M. H. Cheng, "Artificial intelligence enabled routing in software defined networking," Applied Sciences, 10(18), pp. 6564, 2020.
[14] V. Huang, G. Chen, X. Zuo, A. Y. Zomaya, N. Sohrabi, Z. Tari, Q. Fu, "Request dispatching over distributed SDN control plane: a multiagent approach," IEEE Transactions on Cybernetics, 54(5), pp. 3211-3224, 2023.
[15] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, "Modelling the dynamic joint policy of teammates with attention multi-agent DDPG," arXiv preprint arXiv:1811.07029, 2018.
[16] K. I. Qureshi, B. Lu, C. Lu, M. A. Lodhi, L. Wang, "Multi-agent drl for air-to-ground communication planning in uav-enabled iot networks," Sensors, 24(20), pp. 6535, 2024.