Proposing an AI-based model to manage and control congestion at routers on the internet

Vuong Xuan Chi*, Duong Minh Tuan**

Faculty of Information Technology, Nguyen Tat Thanh University
*vxchi@ntt.edu.vn, **dmtuan@ntt.edu.vn

Abstract

The rapid growth of multimedia communication applications drives the demand for network communication, creating significant pressure on network systems and requiring effective queue management solutions to maintain performance and minimize congestion. With fluctuating traffic loads and increasing demands for Quality of Service, traditional queue management methods fail to meet the requirements. To resolve the challenge, the paper proposes the enhanced DAIM-RED model. DAIM-RED offers a comprehensive solution, leveraging adaptive techniques to adjust min-max thresholds, reduce packet drop probability, and optimize queue management efficiency in network routers. The model incorporates AIM-RED, an adaptive method capable of automatically updating the model and tuning parameters based on network data, along with Deep Q-Network, which predicts queue overflow conditions and optimizes throughput. DAIM-RED demonstrates superior network performance compared to models combining AIM-RED with Convolutional Neural Networks and Long Short-Term Memory. The model not only stabilizes queues but also minimizes congestion and ensures Quality of Service in increasingly complex network environments.

Received 20/11/2024
Accepted 09/02/2025
Published 28/02/2025

Keywords

queue management, Deep Q-Network, DAIM-RED, congestion control, network performance

1 Introduction

Nowadays, as the digital age is developing strongly, the need for network communication and data transmission is increasing. Real-time applications, such as online video streaming, video conferencing, and multimedia communication services, are becoming increasingly prevalent [1]. Poor Quality of Service (QoS) affects user experience, especially in real-time applications involving packet delay [2]. The convergence of artificial intelligence (AI) and the internet of things (IoT) achieves significant progress in the industrial sector, with distributed information systems in AIoT specifically designed to address challenges in network environments [3].

https://doi.org/10.55401/y1b92w90

Various research approaches integrate Machine Learning (ML) with the main goal of solving traffic control, congestion control and ultimately improving QoS [4]. Research explores various congestion control and traffic shaping methods, focusing on their application in high-bandwidth, real-time scenarios [5] as well as active queue management (AQM) mechanisms using improved random early detection (RED) algorithms [6, 9]. However, the rapid increase in network traffic places significant pressure on network systems and routing devices, leading to risks of network congestion and reduced performance [9]. A study examines previous research on AQM, identifies challenges arising from AQM-related algorithms in decentralized network configurations, and proposes a comprehensive scheme for new AQM [8]. The development of variants such as Modified RED (MD-RED), which has the ability to randomly adjust the packet drop probability based on the queue length is demonstrated in research [11]. Additionally, an extension to RED called DyRED has been developed to address the limitation and the algorithm was then compared with RED in different network scenarios, to improve network performance and reduce congestion [12]. Improved Active Queue Management (I-RED) implements a combination of non-linear and linear packet drop functions. Performance evaluation I-RED effectively controls the average queue size and delay under light and heavy network traffic conditions [9].

To resolve issue, the authors propose a novel approach based on AI for managing and controlling congestion in network routing systems. Specifically, the authors develop AIM-RED − an enhanced variation of the model predictive control method Improved Random Early Detection (IM-RED) [6], which can automatically adjust control parameters such as threshold values ($\min\_th$, $\max\_th$, $\max_p$). Instead of using fixed parameters, AIM-RED continuously monitors queue state fluctuations and network conditions, updating the IM-RED model and parameters to align with real-time scenarios. This approach minimizes queue overload, reduces unnecessary packet drops, and maintains stable performance under dynamic network conditions.

Additionally, to enhance the ability to predict and manage queue conditions, the authors integrate the Deep Q-Network (DQN) algorithm, a Deep Reinforcement Learning (DRL) method designed to predict queue states and calculate packet drop probabilities. DQN predicts when a queue is likely to become full and determines the optimal packet drop rate. When applied to network routers, DQN continuously observes queue states and forecasts optimal actions to keep queues within safe levels, reduce congestion risks, and ensure stable traffic flow.

In the following sections of the paper, Section 2 − Related Works, describes the RED algorithm, the improved IM-RED algorithm, and the DQN approach. In Section 3, the study presents the DAIM-RED model, including the proposed AIM-RED method and the combined model of AIM-RED and DQN. Section 4 covers the simulation conducted using the NS-2 tool to gather experimental data, followed by experiments with the three models: IM-RED, AIM-RED, and DAIM-RED. Section 5 presents the experimental results and discusses the evaluation, including the comparison of the model with Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). Finally, Section 6 is conclusion and future work.

2 Related Works

2.1 Random Early Detection (RED) Algorithm

RED, proposed by Floyd and Jacobson in 1993, aims to reduce network congestion by dropping packets before the queue becomes full [7]. RED calculates the average queue length and uses it to decide when to drop packets. Many studies improve RED by adjusting the packet drop threshold parameters, such as Adaptive RED (ARED), where the threshold and drop packet probability are adjusted based on the network state. However, these improvements still face challenges when applied in dynamic network environments. Drop probability:

$$P_{\text{drop}}(i) = \begin{cases} 0 & \text{if } q_{\text{avg}}(i) \text{ min\_}th \\ \dfrac{q_{\text{avg}}(i) - \text{min\_}th}{\max_{\text{th}} - \text{min\_th}} \times P_{\max} & \text{if min\_th} < q_{\text{avg}}(i) < \text{ max\_th } q_{\text{avg}}(\text{i}) \\ 1 & \text{if } q_{\text{avg}}(i) \text{ max\_}th \end{cases} \quad (1)$$

Where: q_{avg} represents the average queue value at time i. The $\min\_th$ function is the minimum threshold of the queue. The $\max\_th$ function is the maximum threshold of the queue, and $P_{\max}$ is the maximum packet drop probability.

2.2 Improved Random Early Detection (IM-RED) Method

Many recent studies focus on improving the RED algorithm, addressing issues such as packet loss [13], using quadratic linear methods to enhance queue management [14], and exploring RED with exponential linear functions (RED-LE) [15] which include the interaction of linear loss functions and exponential functions to improve the performance of the original RED algorithm. The TR-RED (Triple-RED) algorithm combines three approaches to address deployment as an alternative to the single linear drop function in RED [16]. Amuel O. Hassan et al. introduced the IM-RED algorithm for congestion control in Internet routers in their study [6]. IM-RED improves the RED algorithm by using two packets drop probability reduction functions, a nonlinear (quadratic) function for light and medium load conditions, and a linear function for heavy load conditions. Research shows that IM-RED reduces the average buffer size and improves latency, especially under heavy load conditions. The advantage of the algorithm lies in its ability to reduce the average buffer size across all load conditions to lower latency, while using a linear function to control congestion more effectively for optimization under heavy load conditions.

In addition, IM-RED is designed based on RED, making it easy to implement on current platforms. IM-RED divides the threshold between $THmin$ and $THmax$ into two ranges, using a quadratic nonlinear function to reduce the packet drop rate and a linear function to increase the packet drop rate. The main steps are:

Compute the average queue size (q_{avg}).

Determine the package drop action based on q_{avg} including the following cases:

– $q_{\text{avg}} < TH_{\min}$: no drop package.

– $TH_{\min} \leq q_{\text{avg}} < Target$: packet drop based on quadratic function.

– $Target \leq q_{\text{avg}} < TH_{\max}$: packet drop based on linear function.

– $q_{\text{avg}} \geq TH_{\max}$: drop the whole package.

Nonlinear (quadratic) decay function:

$$P_b = 9\max_p \times \left(\frac{q_{avg} - TH_{min}}{TH_{max} + TH_{min}}\right)^2 \quad (2)$$

Linear decreasing function:

$$P_b = \max_p + 3(1 - \max_p) \times \left(\frac{q_{avg} - \text{Target}}{2(TH_{max} - 2TH_{min})}\right) \quad (3)$$

Threshold value (Target):

$$\text{Target} = TH_{min} + \frac{TH_{max} - TH_{min}}{3} \quad (4)$$

However, IM-RED has some limitations, such as using two packet drop functions can increase computational complexity and may not be suitable for heterogeneous network scenarios. It requires adjustments to parameters (TH_{min}, TH_{max}, $\max_p$) depending on the specific network conditions.

2.3 Approaching the Deep Q-Network (DQN) Model

Recently, reinforcement learning (RL) methods, particularly DQN, have been applied to address optimization issues in network systems. DQN learns to optimize actions based on the state of the environment, helping to automatically adjust the packet drop probability according to the network's real-time conditions. Features such as packet length, timestamp or transport layer security (TLS) and encrypted payload information can be used as input features to effectively use AI models. Among them, combining deep learning (DL) such as ANN and LSTM with ML algorithms such as RF, KNN LR and SVM, and combining LSTM with CNN can significantly improve the accuracy of the research model [16]. Studies highlight the applications of deep reinforcement learning (DRL), which also examine RL-related algorithms [17], but challenges remain in applying the combination of FRL and deep neural network algorithms for packet prediction in internet networks [0].

To calculate the packet drops probability based on queue forecasting, the DQN method can be used to predict the queue state and optimally adjust the packet drop probability. Approach combines DQN with a queue forecasting model, improving the performance of AQM and minimizing queue overflow. The approach involves:

- Set up the queue model and AQM factors within the DQN environment so the agent can learn how to adjust the packet drop probability to maintain the queue length at an optimal level.
- DQN uses a deep neural network to approximate the Q-value function $Q(s, a; \theta)$. Where, s represents the system state (queue factors), a represents the action (packet drop probability), and θ is a parameter of the neural network (optimized during learning).
- The DQN neural network updates the parameters θ by minimizing the loss function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \quad (5)$$

Where, r is the reward received after performing action a in state s, s' is the new state after action a, γ is the discount factor, which helps balance short-term and long-term rewards, and θ^- is the parameters of the Q-Target network, copied from θ after each step.

- After training, DQN predicts the optimal action (packet drop probability) based on the queue state. When the queue approaches its maximum threshold, packet drop probability increases to reduce the queue load and prevent congestion.

3 Proposing the DAIM-RED Model

3.1 AIM-RED Improvement Method

3.1.1 Model Idea

AIM-RED improves on IM-RED by applying adaptive techniques to automatically adjust the threshold parameters TH_{min}, TH_{max}, $\max_p$ and the packet drop

probability function based on the network state. As a result, AIM-RED reduces the complexity of parameter setting and increases efficiency in heterogeneous networks. It responds quickly to changing network conditions through dynamic parameters and a $sigmoid$ function. The optimization results in improved performance, such as reduced average delay, improved throughput, and a lower unnecessary packet drop rate.

3.1.2 Method Description

- Automatic threshold parameter adjustment:

A dynamic feedback function adjusts $TH_{\min}$, $TH_{\max}$ based on buffer utilization and traffic variation.

Instead of using fixed thresholds, AIM-RED uses dynamic thresholds:

$$TH_{\min}(t) = \alpha \cdot B + (1-\alpha) \cdot U(t) \quad (6)$$

$$TH_{\max}(t) = \beta \cdot B + (1-\beta) \cdot U(t) \quad (7)$$

In which: B is the buffer size, $U(t)$ is the buffer utilization at time t, α and β are the adjustment weights $(0 < \alpha, \beta < 1)$.

- Adaptive packet drops probability function:

A sigmoid function is used instead of a linear or quadratic function to ensure smoother packet drops and more responsive reactions to varying traffic conditions.

$$P_b = \frac{1}{1+e^{-\kappa \cdot (q_{\text{avg}} - \text{Target})}} \quad (8)$$

Where: κ is the amplification factor to adjust the slope of the $sigmoid$ function. The $sigmoid$ function ensures a softer response under light traffic and increases more rapidly under heavy traffic.

- Adaptive update to network conditions:

AIM-RED analyzes $Jitter$ (delay variation) and throughput to adjust dynamic $\max_p$:

$$\max_p(t) = \gamma \cdot (1 - \text{Jitter}(t)) + \delta \cdot \text{Throughput}(t) \quad (9)$$

Where, γ, δ are the weights. $Jitter\ (t)$ is the delay variation at time t, and $throughput\ (t)$ is the throughput at time t.

3.1.3 Proposing AIM-RED Algorithm

Table 1 AIM-RED Improved Method Pseudocoding Algorithm

	Algorithm 1: AIM-RED pseudocode algorithm
1	**Initialize**
2	Buffer size (B)
3	Weighting: α, β, γ, δ
4	Amplification factor: κ
5	$\max_p$ original ($\max_p$ _init)
6	**At each time interval (t)**
7	Calculate buffer usage U_t
8	Update threshold: $TH_{\min}(t)$ and $TH_{\max}(t)$
9	Calculate packet drop probability: P_b
10	Dynamic $maxP$ calculation: $\max_p(t)$
11	If
12	$P_b > \max_p(t)$
13	Drop package
14	**Analysis**
15	Monitor $Jitter(t)$, $Throughput(t)$, and P_b to adjust: $\alpha, \beta, \gamma, \delta$

Table 1 describes the AIM-RED algorithm, which dynamically manages queue thresholds and packet drop probabilities. The algorithm initializes parameters, including buffer size, weighting factors, and amplification factors. At each time interval, it calculates buffer usage, updates min-max thresholds, and determines packet drop probabilities. When the probability exceeds the dynamic maximum value, packets are dropped. The algorithm monitors delay, throughput, and drop probabilities, adjusting parameters to enhance queue management and ensure efficient network operation.

3.2 Proposing DAIM-RED Combination Model

AIM-RED is a dynamic queue management algorithm based on the RED mechanism, enabling better

congestion control and optimizing packet dropping when the queue is full. However, AIM-RED parameters, such as queue thresholds and packet drop probabilities, can be adjusted for even greater effectiveness, and DQN helps find these optimal parameters through the learning process. DQN is a ML method that uses DL in reinforcement learning. Model utilizes neural networks to learn a Q-value function, which helps optimize decisions in dynamic environments like queue management.

3.2.1 DAIM-RED Method

The DAIM-RED model combines AIM-RED with DQN as an improved approach in AQM to optimize network traffic management in routers. Model consists of two main components: AIM-RED, an algorithm automatically adjusts packet drop thresholds based on network conditions, reducing packet loss and delay, and DQN, a model used to determine the optimal action (drop a packet, accept a packet, or adjust AIM-RED parameters) to optimize goals such as reducing delay and increasing network throughput.

The combined approach:

- AIM-RED calculates network metrics, such as queue size, packet loss rate, and average delay, and provides these states as input to DQN.
- DQN learns the optimal policy to select the adjustment parameters for AIM-RED (such as $\min_{th}$, $\max_{th}$, or packet drop rate).
- The action results from DQN are applied to modify the AIM-RED parameters, creating an adaptive optimization loop.

3.2.2 Description of the Proposed DAIM-RED Model

- State:

$$S_t = [Q_t, D_t, P_t] \quad (10)$$

In which, Q_t represents the queue size at time t, D_t is the average delay at time t and P_t is the packet loss rate at time t.

- Action:

$$A_t \in \{ \text{Decrease } \min_{th}, \text{Increase } \min_{th}, \text{Decrease } \max_{th}, \text{Increase } \max_{th} \} \quad (11)$$

- Reward:

$$R_t = -(\alpha \cdot D_t + \beta \cdot P_t) \quad (12)$$

Where, D_t is the average delay, P_t is the packet loss rate, and α, β are the weight coefficients for delay and packet loss.

Action value function (Q-function):

$$Q(S_t, A_t) = R_t + \gamma \cdot \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) \quad (13)$$

Where, γ is the discount factor.

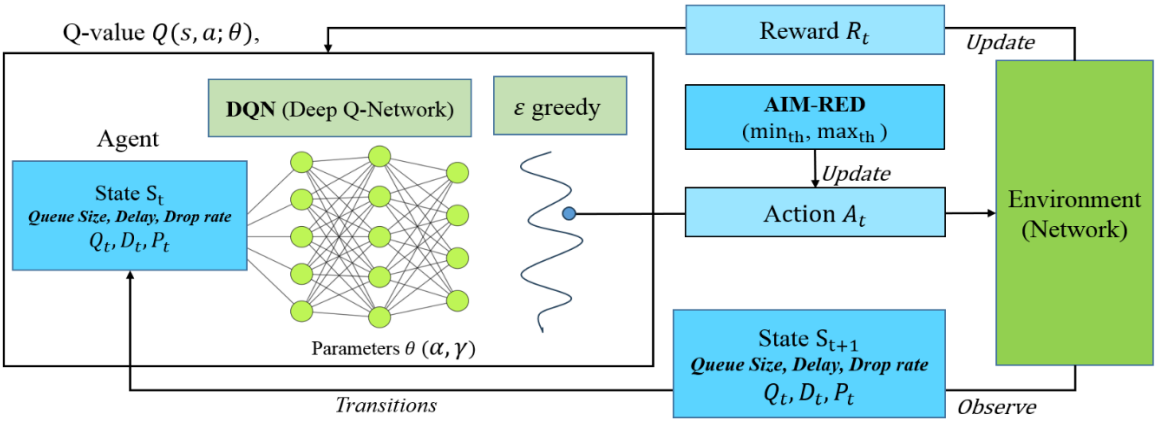

Figure 1 Diagram describing the DAIM-RED Model

Figure 1 includes key components such as the Agent, which receives the current state S_t including parameters such as queue size (Q_t), delay (D_t), and packet loss rate (P_t). DQN uses a deep neural network with parameters θ (α, γ) to compute the Q-value, evaluating the action A_t. First, the agent selects the optimal action A_t or explores a new action based on the $\varepsilon - greedy$. Next, the AIM-RED model adjusts the minimum threshold ($\min_{\text{th}}$) and maximum threshold ($\max_{\text{th}}$) for the RED active queue management algorithm, based on feedback from the DQN network. The network environment then receives the action A_t, returns the new state S_{t+1} and the reward S_t allowing the agent to update its policy.

3.2.3 DAIM-RED Algorithm

The algorithm initializes AIM-RED with default parameters and DQN with a neural network, replay queue, and key parameters. For each episode, it resets the environment, observes states, chooses actions using ε-greedy, updates AIM-RED parameters, trains DQN, and saves optimal models. Table 2 shows the pseudocode algorithm of DAIM-RED.

Table 2 DAIM-RED Model Pseudocode Algorithm

	Algorithm 2: DAIM-RED pseudocode algorithm
1	**Initialize** AIM-RED with default parameters ($\min_{\text{th}}$, $\max_{\text{th}}$, $\max_{\text{p}}$, w_{p}).
2	**Initialize** DQN:
6	*Neural network for function* $Q(S, A)$.
7	*Replay queue.*
8	*Parameters: learning rate* α, *discount factor* γ, ε - greedy
9	**For** each episode:
10	**Reset** the network environment (empty queue, no packet loss)
11	**Get** the initial state: $S_t = [Q_t, D_t, P_t]$
12	**For** each step in the set:
13	Choose action A_t using ε -greedy from $Q(S_t , A_t)$:
14	Take action A_t :
15	Update parameters AIM-RED ($\min_{\text{th}}$, $\max_{\text{th}}$) follow A_t
16	Observe the new state S_{t+1} and the reward R_t
17	Save (S_t, A_t, R_t, S_{t+1}) in replay queue.
18	Train the DQN network by:
19	Get minibatch from replay queue.
20	Update $Q(S_t, A_t)$:
21	**Update** state: $S_t = S_{t+1}$
22	**Save** the best DQN model and AIM-RED parameters

4 Experimentation

4.1 Simulation

The network simulation with NS-2 involves 30 routers connected in a centralized configuration. Routers 1 to 10 are connected to router 11 with a bandwidth of 20 Mbps and a delay of 5 ms. The connection between router 11 and router 12 is a bottleneck connection with a bandwidth of 10 Mbps and a delay of 20 ms. Routers 13 to 30 are connected to router 12, with TCP flows coming from FTP sources and UDP flows from CBR sources, creating bidirectional connections. The window size is limited to 1 000 bytes, and packet drops occur when the queue is full. The simulation runs for 100 seconds.

Figure 2 shows routers routers R11 and R12 are connected with lower bandwidth and higher latency, creating a bottleneck connection that can lead to network congestion. Packet loss also occurs at a point.

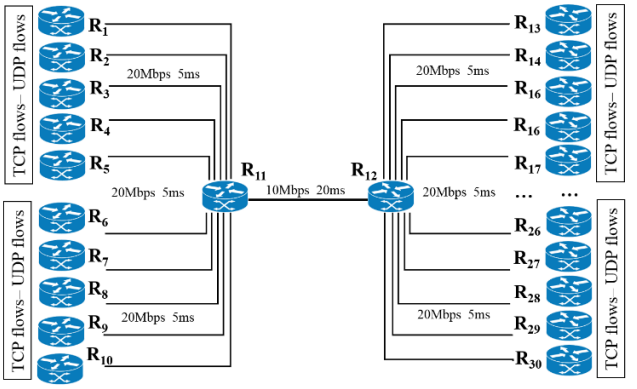

Figure 2 Network Simulation Diagram

The simulation connection is intended to produce uneven and complex data, clearly showing the delay time of packets, allowing for more accurate analysis and test evaluation. The total simulation time is 100 seconds, with packet traffic monitored to obtain test data. The time period is chosen because it is long enough to simulate real network situations and keep the data volume manageable, allowing for more efficient analysis and processing.

```
RangeIndex: 1133916 entries, 0 to 1133915
Data columns (total 20 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   event_type      1133916 non-null  object
 1   time            1133916 non-null  float64
 2   src_node        1133916 non-null  int64
 3   dst_node        1133916 non-null  int64
 4   packet_type     1133916 non-null  object
 5   packet_size     1133916 non-null  int64
 6   flags           1133916 non-null  object
 7   fid             1133916 non-null  int64
 8   src_ip          1133916 non-null  float64
 9   dst_ip          1133916 non-null  float64
 10  seq_num         1133916 non-null  int64
 11  packet_id       1133916 non-null  int64
 12  d_processing    1133916 non-null  float64
 13  d_queue         1133916 non-null  float64
 14  d_transmission  1133916 non-null  float64
 15  d_propagation   1133916 non-null  float64
 16  average_delay   1133916 non-null  float64
 17  total_delay     1133916 non-null  float64
 18  queue_size      1133916 non-null  int64
 19  queue_label     1133916 non-null  object
dtypes: float64(9), int64(7), object(4)
memory usage: 173.0+ MB
```

Figure 3 Network Data after Processing

The simulation result shows the generated packets have 12 fields, including the IP-header fields. From the trace data in the simulation, after analyzing and processing the data, the delay, queue length, and packet drop rate are calculated. The simulation provides data for experimentation. Figure 3 shows a total of 1,133,916 packets transmitted over the network, and the processed data contains 20 attributes.

4.2 Model Experiment

The experiments evaluate the effectiveness and adaptability of queue management models (IM-RED, AIM-RED, and DAIM-RED) under various network conditions.

4.2.1 Experiment 1: Performance Evaluation with Fixed Parameters

Fixed parameters such as min_{th}, max_{th}, and max_p aim to evaluate the performance of the IM-RED model:

– min_{th} = 10 packets (minimum queue size threshold).

– max_{th} = 30 packets (maximum queue size threshold where the marking/drop probability reaches 100%).

– max_p = 0.2 (maximum packet marking probability).

– w_q = 0.002 (weight factor).

The experiment measures the average queue size to determine the model's effectiveness in maintaining acceptable delay levels. It calculates packet drop probability, queue size, and delay to analyze queue behavior over time in detail.

4.2.2 Experiment 2: Evaluating Adaptability to Network Conditions

In the AIM-RED model, the thresholds min_{th}, max_{th}, and max_p dynamically update based on the current network state (queue size, delay, and packet loss rate):

– min_{th} = (5 to 20) packets (changes with network conditions)

– max_{th} = (30 to 80) packets (changes with network conditions)

– max_p = (0.05 to 0.5) (changes with network conditions)

The experiment evaluates queue utilization efficiency, the ability to reduce packet loss, and throughput improvement. It also assesses queue stability and network performance as the minimum and maximum thresholds change. Approach analyzes the adaptability and effectiveness of the model in fluctuating network conditions.

4.2.3 Experiment 3: Optimizing packet drop probability and threshold

The optimization and learning capability of DAIM-RED integrates reinforcement learning techniques (DQN) with AIM-RED, using the following parameters:

$state\_size$ represents the number of input states, $action\_size$ represents the number of actions the agent can take, and the agent stores experiences, including state, action, reward, and next_state, for training the DQN network.

– γ = 0.95 (discount factor).

– ε = 1.0 (exploration factor for discovering new actions).

– $\varepsilon_{\min}$ = 0.01 (minimum value of epsilon as exploration decreases).

– $\varepsilon_{\text{decay}}$ = 0.995 (rate of epsilon decay over time).

– size = 32 (batch size for stabilizing the optimization process during training).

The DQN approach learns optimal actions, balancing exploration and exploitation to achieve high performance in complex and dynamic network conditions. The goal is to optimize thresholds and packet drop probabilities through reinforcement learning.

Additionally, the research experiments with other AI models, such as CNN and LSTM combined with AIM-RED. CNN excels in processing spatially structured data, such as images or data matrices. When integrated with AIM-RED, CNN effectively analyzes features like bandwidth, latency, and queue size through matrix representations. LSTM serves as an optimal choice for handling sequential data, such as time series of network traffic. Combined with AIM-RED, LSTM enables the model to learn patterns of delay and dynamic changes in network traffic, ensuring better predictions for time-dependent data.

5 Results, Evaluation, and Discussion

5.1 Results

5.1.1 Experiment Results 1

The IM-RED model represents parameters such as packet drop rate (%), average delay, average queue size, and network throughput in the experimental model over 100 seconds network simulation run time.

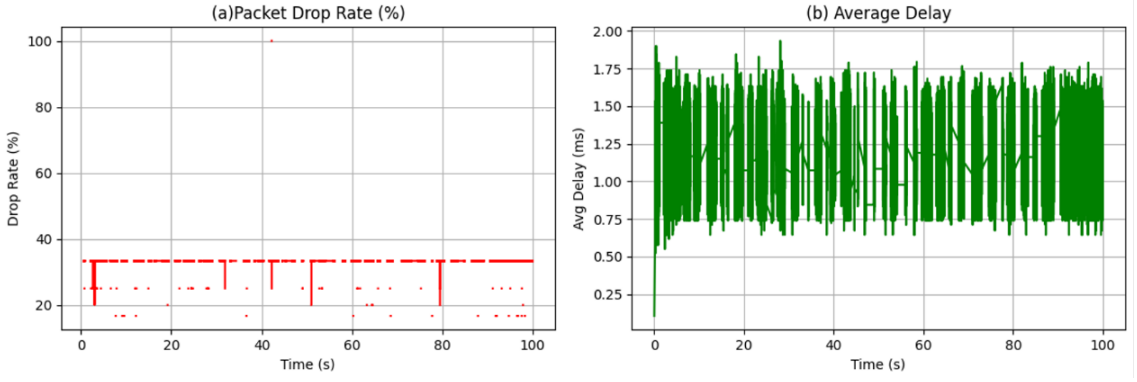

Figure 4 (a) Packet Drop Rate and (b) Average Delay of IM-RED Model

The Packet drops rate chart, calculated as a percentage (%), reflects the queue control capability of the mechanism. A high drop rate indicates many packets are lost due to queue overflow. As shown in Figure 4a, the drop rate varies between 0% and over 35%, indicating a relatively high level. Figure 4b illustrates the average packet delay in the network, encompassing processing, queueing, transmission, and propagation times. The average delay ranges from 0.75 ms to over 1.75 ms.

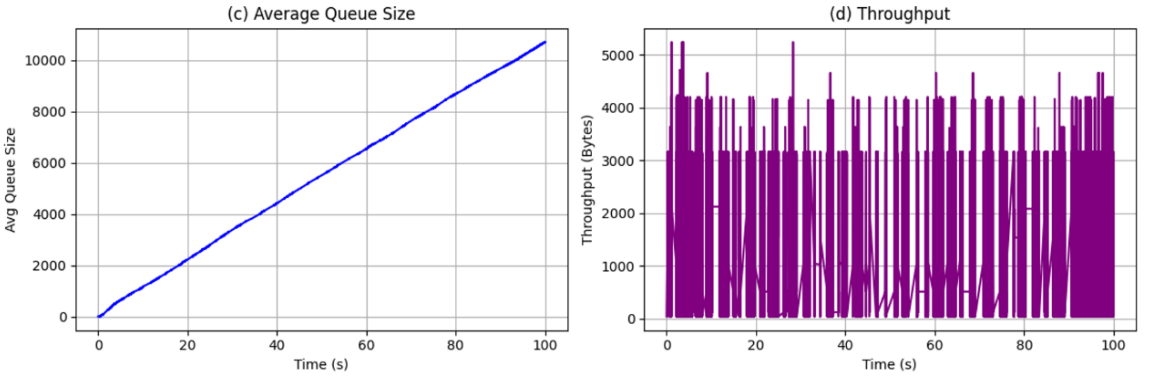

Figure 5 (c) Average queue size and (d) throughput of IM-RED model

Figure 5c illustrates the Average Queue Size measured in the number of packets, indicating the queue size gradually increases from 0 to over 10,000 packets within 100 seconds, highlighting a serious queue buildup. Figure 5d shows throughput measured in Mbps, reflecting the amount of data processed and transmitted through the network over time. Throughput fluctuates between 0 Mbps and over 0.04 Mbps.

5.1.2 Experiment Results 2

Experiment result 2 shows the changes in the metrics when the proposed AIM-RED method is applied.

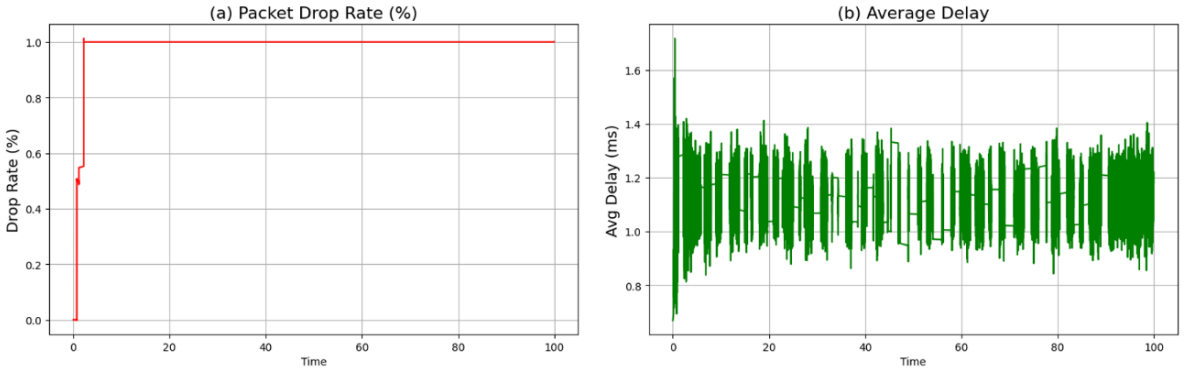

Figure 6 (a) Packet Drop Rate and (b) Average Delay of AIM-RED Model

Figure 6a presents the packets drop rate decreases significantly, fluctuating from 0% to around 1% and stabilizing by the 100th second. Figure 6b shows the Average Delay, which remains stable at a lower level, fluctuating between 0.9 ms and 1.4 ms.

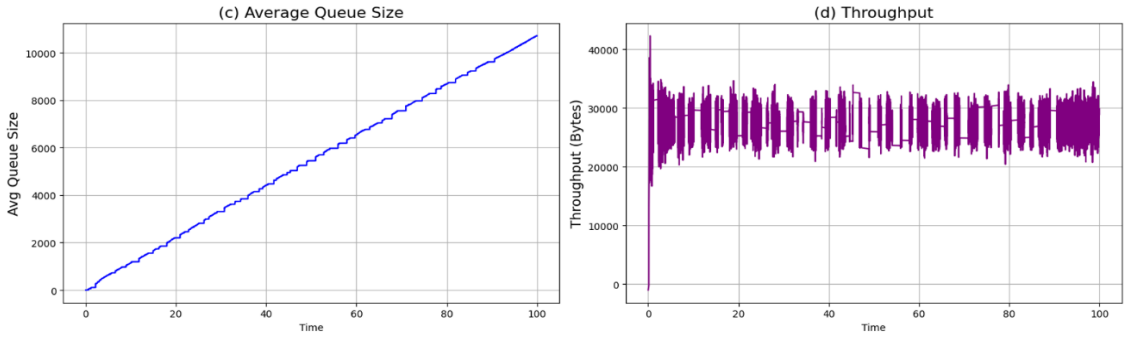

Figure 7 (c) Average Queue Size and (d) throughput of AIM-RED Model

Average queue size plot in packets. Figure 7c illustrates the queue size fluctuates less, similar to experiment 1, and the throughput in Figure 7d remains higher, ranging from over 20 Mbps to over 30 Mbps.

5.1.3 Experiment Results 3

The experiment with the DAIM-RED model shows the combination of the DQN algorithm with AIM-RED to solve the reinforcement learning problem. DQN is used to train an agent to perform actions based on the current state of the environment to optimize long-term cumulative rewards. The results achieve better performance than before the combination when tested over 100 episodes.

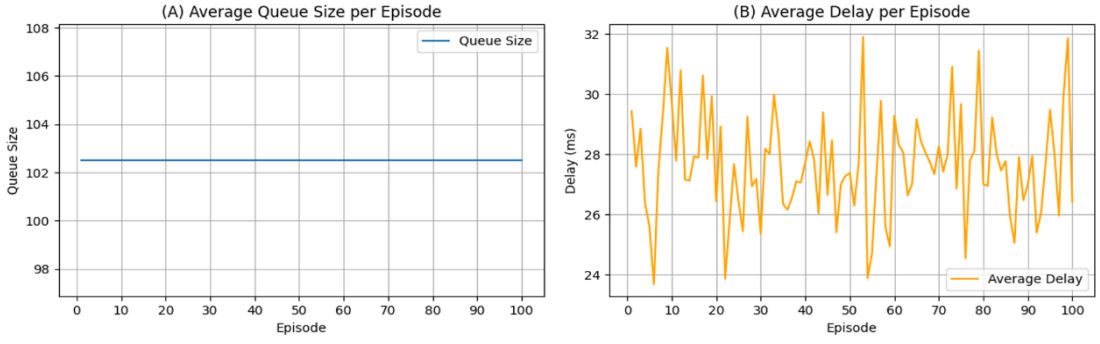

Figure 8 (a) Average Queue Size and (b) Average Delay when Executing 100 Episodes

Figure 8a shows the average queue size remains stable at around 102-104 throughout the 100 episodes. Figure 8b illustrates the average delay fluctuates between 24 ms and 32 ms across the episodes. The graph shows significant fluctuations in some episodes, but there is no notable increasing or decreasing trend.

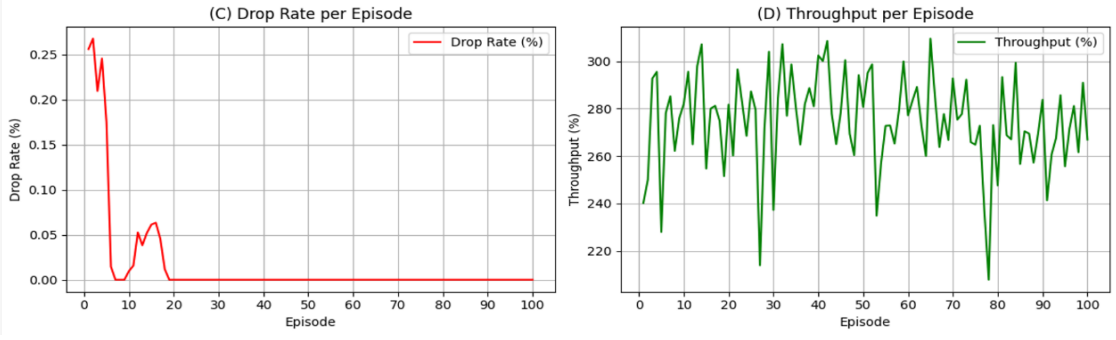

Figure 9 (c) Packets Drop Rate and (d) throughput Evaluation when Performing 100 Episodes

Figure 9c presents the packet drop rate initially starts high at 25% but drops significantly to near 0% after the first 10 episodes and remains stable until the 100th episode. Meanwhile, the throughput fluctuates between 220% and 300% across the episodes, showing no declining trend but displaying some strong fluctuations, as shown in Figure 9d.

5.2 Evaluation and Discussion

The charts above demonstrate the DAIM-RED model operates effectively in maintaining stable queue sizes, reducing packet drop rates, and keeping throughput high. However, there is significant delay variation, which negatively impacts the quality of service (QoS), especially in applications requiring low latency. IM-RED (Experiment 1) shows the lowest performance among the three models, with instability and suboptimal QoS metrics. AIM-RED (Experiment 2) presents improvements but still exhibits instability in some metrics such as delay and packet drop rate. DAIM-RED (Experiment 3) illustrates significant improvements across all QoS metrics, particularly with packet drop rates nearly reaching 0% and stable throughput. Notably, the integration with DQN helps the system learn and optimize performance over time. Table 3 shows a comparison of the models.

Table 3 Comparison and Evaluation Table of IM-RED, AIM-RED, and DAIM-RED Models

Evaluation criteria	**IM-RED**	**AIM-RED**	**DAIM-RED**
Average Queue Size	Large oscillation from 0 to 200, unstable.	More stable, but still some fluctuations.	Stable at 102-104, no big fluctuations.
Average Delay	Oscillation from 0 ms to 50 ms, high volatility.	Volatility decreased, but still high.	More stable, small fluctuations in the range of (24-32) ms.
Packet Drop Rate	Initially high, gradually decreasing but not stable.	Significantly decreased, but still some spikes in some periods.	Rapidly drops from 25% to nearly 0% after only 10 episodes, maintaining stability.
Throughput	Oscillation is not yet unstable.	Improved, less fluctuations, but not yet highly stable.	More stable, maintaining high throughput in the range of 220%-300%.
Learning Ability	None.	None.	Yes, using DQN to learn and optimize actions, improving efficiency.
Improvement Level	Improved from traditional RED, but not fully optimized.	Adjusted from IM-RED, optimized, but not yet complete.	Outstanding improvement with DQN combination, comprehensive QoS optimization.

The experimental results evaluate the performance of DAIM-RED compared to CNN-AIM-RED and LSTM-AIM-RED. The evaluation measures the accuracy and loss of the models and visualizes the metrics through charts for Train Accuracy, Train Loss, Validation Accuracy, and Validation Loss.

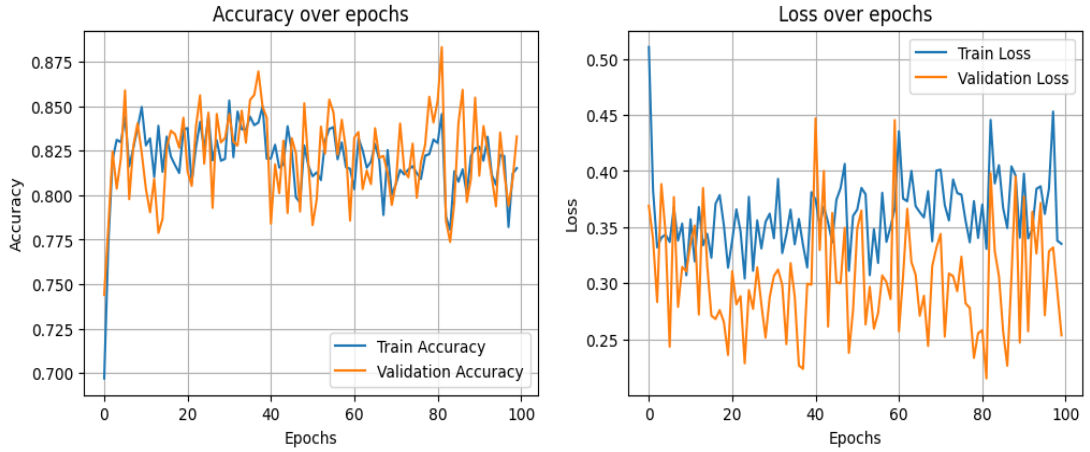

Figure 10 Accuracy and Loss of the CNN-AIM-RED Model over 100 Epochs

Figure 10 depicts the accuracy and loss of the CNN-AIM-RED model. Train accuracy remains stable around 85%-87%, and validation accuracy peaks at nearly 88%, though it remains lower than DAIM-RED. Train loss decreases steadily, and validation loss shows greater stability compared to LSTM-AIM-RED but remains higher than DAIM-RED. While it demonstrates better stability than LSTM-AIM-RED, it does not reach the performance level of DAIM-RED.

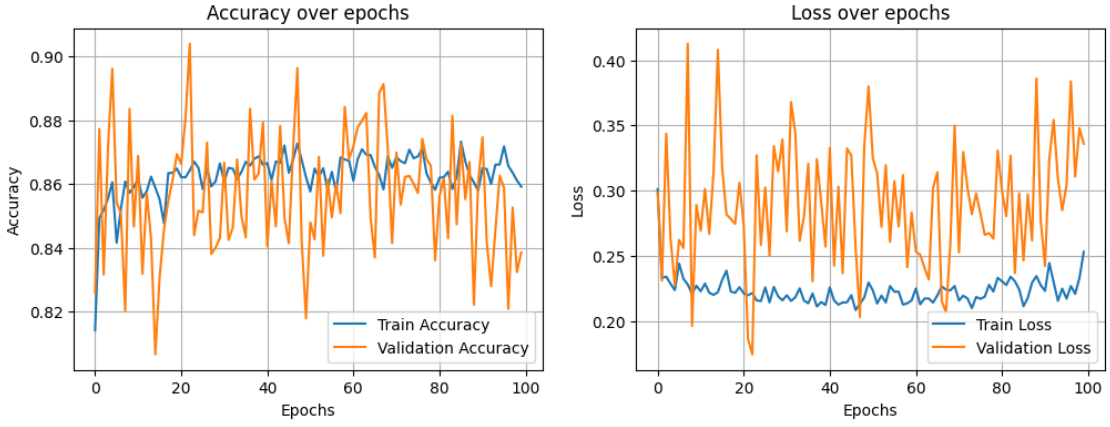

Figure 11 Accuracy and Loss of the LSTM-AIM-RED Model over 100 Epochs

Figure 11 illustrates the accuracy and loss of the LSTM-AIM-RED model. Train accuracy reaches around 85%, but validation accuracy fluctuates significantly, ranging from 82% to 90%, with lower stability compared to DAIM-RED. Additionally, train loss decreases steadily, but validation loss shows large fluctuations, indicating weaker

generalization ability. The model handles sequential data effectively but demonstrates less adaptability to changes in the network compared to DAIM-RED.

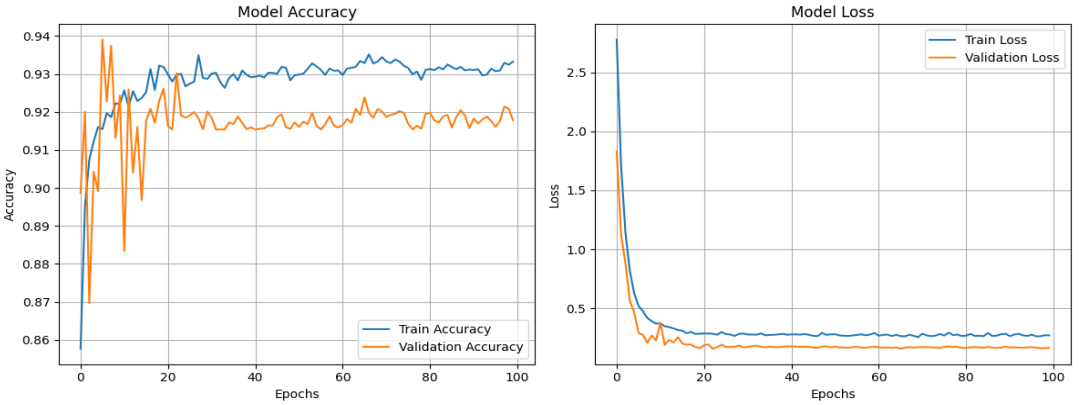

Figure 12 Accuracy and Loss of the DAIM-RED Model over 100 Epochs

In Figure 12, the accuracy and loss of the DAIM-RED model indicate that its Accuracy reaches the highest level among the three models, with validation accuracy consistently exceeding 93%. It shows less fluctuation compared to LSTM-AIM-RED. Both train loss and validation loss decrease rapidly and remain stable, demonstrating that DAIM-RED generalizes better than the other two models.

Table 4 Comparison and Evaluation Table for the CNN-AIM-RED, LSTM-AIM-RED and DAIM-RED Models

Model	Accuracy (%)	Loss (%)	Dynamic features	Stability
CNN-AIM-RED	85-87	24-35	Limitations in sequential data	Stable but not high
LSTM-AIM-RED	82-89	23-37	Handles temporal data well	Less stable than DAIM-RED
DAIM-RED	91-94	16-18	Adapts well to change	Very stable

Table 4 presents the metrics of the DAIM-RED model, showing an approximately 7% higher Accuracy compared to the CNN-AIM-RED model and around 5% higher than the LSTM-AIM-RED model. Meanwhile, the loss of the DAIM-RED model is lower than that of both the CNN-AIM-RED and LSTM-AIM-RED models. The DAIM-RED model utilizes a DQN network to optimize queue management in AQM. The model applies reinforcement learning to identify an optimal policy, reducing average latency and adapting effectively to changes in data flow.

6 Conclusion and Future work

The proposed DAIM-RED model combines the enhanced AIM-RED method with DQN to optimize queue state prediction and determine packet drop probabilities. The integration of AIM-RED and DQN provides a comprehensive approach, utilizing adaptive techniques to handle complex network states, thereby optimizing traffic control and improving QoS in network systems. Compared to the CNN-AIM-RED and LSTM-AIM-RED models, DAIM-RED achieves high training accuracy, offering flexibility and adaptability to dynamic and diverse network

environments, effectively enhancing congestion management and control.

The practical development of the DAIM-RED model focuses on applications in large-scale networks, particularly 5G, IoT, and cloud data centers. Integrating DAIM-RED into routing devices can improve queue management and optimize network traffic in communication networks. Furthermore, the model can be extended to support more complex networks, such as Software-Defined Networking (SDN) or Network Function Virtualization (NFV), to enhance automation and system performance. Future research may focus on optimizing the model, reducing computational overhead, and increasing its applicability in mitigating congestion at routers.

Acknowledgment

This research is funded by Nguyen Tat Thanh University, Ho Chi Minh City, Viet Nam for Science and Technology Development under grant number 2025.01.02/HĐ-KHCN

References

1. Barakabitze, A. A., Barman, N., Ahmad, A., Zadtootaghaj, S., Sun, L., Martini, M. G., & Atzori, L. (2019). QoE management of multimedia streaming services in future networks: A tutorial and survey. *IEEE Communications Surveys & Tutorials*, *22*(1), 526-565.2.

2. Usmanova, N. B., Mirzayev, D. A., Ergashev, F. A., & Yunusova, D. A. (2023). Field monitoring application based on video surveillance: Evaluation of system performance. *In E3S Web of Conferences* (Vol. 443, p. 06016). *EDP Sciences.*

3. Xie, Z., Ji, C., Xu, L., Xia, M., & Cao, H. (2023). Towards an optimized distributed message queue system for AIoT edge computing: a reinforcement learning approach. *Sensors*, *23*(12), 5447.

4. Panahi, P. H., Jalilvand, A. H., & Diyanat, A. (2024). Enhancing Quality of Experience in Telecommunication Networks: A Review of Frameworks and Machine Learning Algorithms. *arXiv preprint arXiv*:2404.16787.

5. Velayutham, A. (2022). Congestion control and traffic shaping in high-bandwidth applications: Techniques to manage network congestion and optimize traffic flow in gaming, ar/vr, and cloud services. *Eigenpub Review of Science and Technology*, *6*(1), 144-165.

6. Hassan, S. O., Rufai, A. U., Agbaje, M. O., Enem, T. A., Ogundele, L. A., & Usman, S. A. (2022). Improved random early detection congestion control algorithm for internet routers. *Indonesian Journal of Electrical Engineering and Computer Science*, *28*(1), 384-395.

7. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, *1*(4), 397-413.

8. Stoltidis, A., Choumas, K., & Korakis, T. (2024, January). Active Queue Management in Disaggregated 5G and Beyond Cellular Networks Using Machine Learning. *In 2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)* (pp. 113-120). IEEE.

9. Samuel Oluwatosin Hassan, et al. (2021). I-RED: An Improved Active Queue Management Algorithm. *Journal of Computer Science. 18*(3): 130.137. http://dx.doi.org/10.3844/jcssp.2022.130.137

10. Nasiri, B., Bayat, F., Mohammadkhani, M., & Bartoszewicz, A. (2024). Optimal congestion management in network routers subject to constraints, disturbances, and noise using the model predictive control approach. *IET Cyber-Physical Systems: Theory & Applications*, *9*(3), 258-268.

11. Hassan, S., & Rufai, A. (2023). Modified dropping-random early detection (MD-RED): a modified algorithm for controlling network congestion. *International Journal of Information Technology*, *15*(3), 1499-1508.

12. Danladi, S. B., & Ambursa, F. U. (2019, December). DyRED: An enhanced random early detection based on a new adaptive congestion control. In *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)* (pp. 1-5). IEEE.

13. Abu-Shareha, A., Al-Kasasbeh, B., Shambour, Q. Y., Abualhaj, M. M., Alsharaiah, M. A., & Al-Khatib, S. N. (2022). Linear random early detection for congestion control at the router buffer. *Informatica*, *46*.

14. Hassan, S. O., Nwaocha, V. O., Rufai, A. U., Odule, T. J., Enem, T. A., Ogundele, L. A., & Usman, S. A. (2022). Random early detection-quadratic linear: an enhanced active queue management algorithm. *Bulletin of Electrical Engineering and Informatics*, *11*(4), 2262-2272.

15. Hassan, S. O. (2022). RED-LE: A revised algorithm for active queue management. *Journal of Telecommunications and Information Technology*.

16. Samuel O. Hassan, et al. (2023). A Triplex Region-Random Early Detection (TR-RED) Algorithm for Active Queue Management in Internet Routers. *International Journal of Computing and Digital Systems*. *14*(1) 193-201 (Jul-23). http://dx.doi.org/10.12785/ijcds/140117

17. Elmaghraby, R. T., Aziem, N. M. A., Sobh, M. A., & Bahaa-Eldin, A. M. (2024). Encrypted network traffic classification based on machine learning. *Ain Shams Engineering Journal*, *15*(2), 102361.

18. Wang, Q., & Tang, C. (2021). Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowledge-Based Systems*, *233*, 107526.

19. Matrood, M. Q., & Ali, M. H. (2024). Enhancing Network Congestion Control: A Comparative Study of Traditional and AI-Enhanced Active Queue Management Techniques. *Journal of Cybersecurity & Information Management.*

Đề xuất mô hình dựa vào trí tuệ nhân tạo để quản lý và điều khiển tắc nghẽn tại bộ định tuyến trên mạng truyền thông Internet

Vương Xuân Chí*, Dương Minh Tuấn**
Khoa Công nghệ Thông tin, Trường Đại học Nguyễn Tất Thành
*vxchi@ntt.edu.vn, **dmtuan@ntt.edu.vn

Tóm tắt Sự phát triển mạnh mẽ của các ứng dụng truyền thông đa phương tiện đã làm tăng nhu cầu giao tiếp mạng, gây áp lực lớn lên hệ thống mạng và đòi hỏi các giải pháp quản lý hàng đợi hiệu quả để duy trì hiệu suất, giảm thiểu tắc nghẽn. Trước sự biến động của lưu lượng truyền tải và yêu cầu ngày càng cao về chất lượng dịch vụ (QoS), các phương pháp quản lý hàng đợi truyền thống không còn đáp ứng được. Để giải quyết vấn đề này, bài báo đề xuất mô hình cải tiến DAIM-RED, mang lại giải pháp toàn diện, sử dụng kỹ thuật thích nghi để điều chỉnh ngưỡng min-max, giảm xác suất thả gói và tối ưu hóa hiệu quả quản lý hàng đợi tại các router mạng. Mô hình bao gồm AIM-RED, một phương pháp thích nghi có khả năng tự động cập nhật mô hình và điều chỉnh tham số dựa trên dữ liệu mạng, cùng với Deep Q-Network, giúp dự đoán tình trạng hàng đợi đầy và đưa ra tỷ lệ bỏ gói tin tối ưu. DAIM-RED có hiệu năng mạng tối ưu hơn so với mô hình kết hợp AIM-RED với CNN và LSTM. Mô hình không chỉ đạt hiệu năng mạng cao hơn mà còn giảm thiểu tắc nghẽn, đảm bảo QoS trong môi trường mạng ngày càng phức tạp.

Từ khóa quản lý hàng đợi, mạng học sâu tăng cường, DAIM-RED, kiểm soát tắc nghẽn, hiệu năng mạng