Al-Salam Journal for Engineering and Technology
Journal Homepage: http://journal.alsalam.edu.iq/index.php/ajest
e-ISSN: 2790-4822 p-ISSN: 2958-0862

Adaptive Random Early Detection Using Deep Reinforcement Learning for Enhanced Congestion Control in Dynamic TCP/IP Networks

Mohammed Q. matrood[1]*, majid H. ali[2]

[1]Anbar Education Directorate, Anbar Governorate, Ministry of Education, Iraq.
[2]Department of Computer Science, Computer Science and Mathematics College, Tikrit University, Tikrit, Iraq.

*Corresponding Author: Mohammed Q. matrood

DOI: https://doi.org/10.55145/ajest.2025.04.01.013
Received August 2024; Accepted October 2024; Available online November 2024

ABSTRACT: AQM is used to meet targets for congestion control and low delay, high throughput in TCP/IP networks. But methods developed for AQM like Random Early Detection (RED) work strictly on the control parameters pre-set on them and thus may not respond very well to changing network conditions. Thus, this paper presents a composite AQM model using DRL based on DQN, and the ability to learn independently regarding the AQM weight parameter of the queue. The fact that DRL is adaptive means the effectiveness of the proposed system is also assured. As it is observed in implementing traditional model-based approach, the stability and performance degrade when tested in different network conditions; however, DRL's ability to learn independently enables ML to solve network congestion as it happens. Consequently, more stability is achieved, the delay is lesser, more bandwidth is used and the overall packet drop rate is low; the proposed DRL-RED model is ideal for dynamic network environments. The proposed DRL-RED model is compared with the regular RED algorithm for both low density and high-density networks. This proposed model has achieved sustained throughput of up to 49.9 Mbps with 0.949 % reduction on delay and very low PLR of 8.38043%. From the comparative analysis, it is clear that the employment of DRL with RED improves the stock of network throughput, the reduction on packet drop frequency, and the general delay in network situations. There are two main disadvantages of this approach: first, it cannot reach the heavy-traffic, the problem partly solved by model-based approach; second, the values of the congestion-control parameters cannot be reset, an issue eradicated by DRL as a result of its inherent adaptiveness. Consequently, new stability and increased performance can be achieved under various network conditions.

Keywords: Congestion Control, Random Early Detection, Deep Reinforcement Learning

1. INTRODUCTION

As the size of the network increases, the problem of congestion control becomes a major issue and high-priority Issue [1]. This paper focuses on the TCP approach to congestion control, as it is one of the most widely used Internet protocols for traffic and congestion management, whose effectiveness determines Internet speeds. TCP uses a congestion control technique at the transport level so that the network is always efficient and that the sender doesn't flood the network. With rising bit rates in data communication, congestion, when it occurs, must either be avoided or reduced in degree [2].

In particular, traditional TCP congestion control was designed almost entirely for long-lived connections and does not perform optimally for short-lived traffics such as Web, Cloud Computing and Internet of Thing (IoT). RED assist in controlling buffer sizes but implementing elaborate methods like Deep Q-Networks (DQN) could assist in enhancing the working of AQM in fast networks [3]. Current and future applications like Voice over IP (VoIP), video conference, real time video streaming, email and file transfer have dissimilar quality of service (QoS) demands on the communication and computing networks. Real-time services that include VoIP and video and live streaming services for instance, demand considerable bandwidth besides having little or no tolerance to both latency and jitter. The good

thing with application for networks are typically more robust to packet loss [4]. TCP has congestion control, but the congestion that occurs within a router's buffers when TCP is used across networks is not managed by TCP. Congestion sharply degrades the networks' performance through escalating the delay in queues and packets loss, and, decreases packets delivery rate, or the throughput [5, 6]. In attempting to tackle these problems many Active Queue Management (AQM) algorithms have been introduced as they help to identify congestion early and improve the networks' performance. Some of them are BLUE [7], Proportional Integral Controller Enhanced (PIE) [8] and RED [9] etc.

Other AQM algorithms such as RED estimates congestion based on queue size [10], and uses a random based technique to proactively drop packets to prevent congestion. BLUE has been acknowledged for its technique in which packets are dropped randomly to contain congestion; this improves the nature of packet rejection probability but weakens the stability of the network. In turn, this is why Proportional Integral controller Enhanced (PIE) refines this approach by altering it depending on the changes in queue length. Nevertheless, all the AQM algorithms, RED, PIE, and BLUE, utilize fixed network setting and parameters that require to be set locally. This may challenge the model's congestion control because congestion parameters and their density in different congestion levels increases the model's volatility and potential errors.

When applied to the gateway level, it is absolutely beneficial as it ends emulator dreams and solves buffer bloat and congestion in TCP/IP networks. The Bogus RED algorithm discards packets even before the buffer occupancy attains its limit while PIE and BLUE present derivatives of this idea. Congestion usually occurs when the delivery of packets in the network is higher than the channel rate of the bottleneck link. During congestion, each packet ends up in a buffer and may be dropped depending on the availability of buffer and length of the queue. It can be seen that system through put and the bottleneck channel capacity can be represented by $PX=B$. However, buffer bloat normally interferes with this free-flowing process. An important goal of AQM is to reduce queue length and improve the balance of the network that is a part of the main intuitions behind it [11].

Deep Reinforcement Learning (DRL) provides a model free solution to get the best tuning in complex and dynamic network scenarios [12]. This article explains that within the reinforcement learning environment DRL it is possible to learn to optimize the weight queues (wq) over time, without necessarily needing to understand the network architecture.

The objective of this research is to enhance TCP congestion control via DRL using the Deep Q-Network (DQN) to ascertain precise values of wq parameters. Thus, the goal of the study is to eliminate packet losses and avoid such impacts on TCP congestion management as global synchronization and oscillations, using DRL to improve TCP's performance end-to-end to maintain the quality of the user experience at a high level. This is attained through estimating and learning the probability density of packets arrival. Like any algorithm, the DRL algorithm operates based on the parameters, where a change in one parameter affects others; the model-free approaches develop an emphasis on congestion and enhance the AQM algorithms' efficiency. DRL has been applied before in deal with congestion in various fields such as networking, robotics, and Internet of Things (IoT) among others [12].

Being an advanced model, the DRL model mitigates fluctuation typical to the use of methods such as RED in dynamic and high-density mesh networks [12, 13]. The main objective of this work is to increase efficiency by adopting an adaptive model based on DQN. It also got the asset of influencing the flexibility and adaptability of the network to enable it handle the communication flows. The congestion management system of the proposed framework is an AQM congestion control system, which comprises PD controller, DOB, and SP. Subsequent simulation tests conducted in ns-2 verified the method because the proposed approach of PD + DOB + SP demonstrated better throughput than the traditional approach, including cases where UDP flows mix with other traffic [14].

The proposed DRL-AQM algorithm also brought up a model-free approach based on DRL strategies. DRL-AQM succeeds in determining the best packet-dropping policies from the traffic models they learn; therefore, DRL-AQM can improve the performance of different network conditions. Experiments prove that DRL-AQM is robust and self-learning to outcompete other conventional AQM strategies under different challenging scenarios of network connections through adapting to dynamic link conditions [10].

Moreover, the usage of the RED with Reconfigurable Maximum Dropping Probability (RRMDP) method also improves the network performance by tuning the maxp value according to traffic condition. This dynamic adjustment is most appropriate for controlling average queue size (avg) while preventing excessive queuing delay with no effect on the drop rate or overall link usage [15]. To further reduce congestion in TCP/IP networks, this research novelly proposes a weighted ensemble DRL model with five algorithms: Twin Delay DDPG, Proximal Policy Optimization, DQN, and DDPG. It compared well with other architectures and congestion control algorithms such as DRL and RED and proved superior to single DRL models during normal operations and specifically under conditions that were stressful [14].

Reinforcement learning (RL) has received a lot of attention from researchers as a result of its potential to improve throughput and reduce network latency. So the research proposes DRL for enhancing the AQM system. The DRL is capable of handling any sophisticated network with intensive variables and increases the power of the DRL model. Using the parameters, this study creates a DRL for AQM congestion management. In this paper we use the DQN algorithm to select better wq, to enhance throughput, delay, and packet loss rate (PLR).

The contributions of the paper are:

• Simulate a dumbbell TCP network using normal and stress testing to account for fluctuations in traffic that arrives at the router.

• Working on finding the best waiting weight queue by using the DQN algorithm which ensures high throughput, less packet loss, and delay in AQM.

Section 2 covers DRL techniques for AQM control congestion research and publication. Section 3 discusses AQM and DRL-RED. Section 4 describes the approach taken to explain the study's DRL models. Section 5 discusses simulations and assessment of the setting. Finally, we provide our conclusions.

2. RELATED WORKS

In [10], DRL-AQM deals with the network congestion especially in data transmission congested areas, which is a common problem. DRL-AQM methods are used most often to solve the problem of congestion in the network. This paper analyses the main sources and consequences of network congestion with especial emphasis at buffer bloat. In AQM systems, adaptive DRL has been observed to enhance throughput variability together with the enhancement of delays. In a less complex network context, offered a method in which queue wait times are cut by as much as 400 while sustaining normal throughput rates. PPO algorithm from RL is used for accomplishing policy learning In particular, it is claimed that PPO keeps stable the throughput.

In Smart Queue Management in [12], decide to use DQN because it is effective in reaching throughput and latency goals. A new AQM method is implemented in fog and edge networks in [16] to minimize queuing delay where DRL is used to upsurge queuing latency impartially to throughput. This approach was compared with other AQM methods including P-FIFO, RED, and CoDel all of which were outperformed by the proposed method in series of tests that yielded lower latency and jitter yet higher than average throughput. The states of the system include packet queue length, queue rate, and queue delay while actions include packet dropping and forwarding.

In the work [17], deep neural network is used to decide dynamically about the buffer size of each network flow depending on congestion, latency, and bandwidth demands. This strategy makes it possible for the system and developers to get through with lesser queues while still having an allowance for throughput. As explained in [18], DQN can be employed in managing traffic for Multi-Path TCP (MPTCP) in having a DQN agent select the best routes and showing its ability in handling problems in traffic management and selection of paths. Furthermore, [19] discusses the application of RL to optimize MPTCP flow Scheduling for short and long flows with superior performance to the method that has already been used.

The proposed DDPG-based DRL system will use the path dynamic packet distribution where the goal is to reduce the number of out of order queues in each path. The approach taken employs an actor-critic method, with transformers used for processing the change in states of sub-flows of neural networks. In Reference [20], a DRL algorithm for the network slice resource management is presented to consider fluctuations commonly observed in traffic arrival patterns and resource requirements which may include computational, memory, and bandwidth resources. As this method proves that its better in terms of resource utilization and has less latency compared to equal slicing approach.

In [21] as the study of the DQN algorithm, two types of rewards are used including delayed rewards with an increase in delay decrease and enqueue rewards which increase with the decreasing packet drop rates. These rewards must be properly normalized; for instance, if packet drops are frequent, the delayed reward goes up and if most packets are put in the queue, the enqueue reward also goes up, possibly to a level that increases the delay. Achievement of this balance is made possible through the use of scaling parameter, δ. Also, the element of size for training data enhances the algorithm output delivered by the program. The test result showed that the Proposed DRL-based AQM had better performance than RED in latency and system throughput across the different test scenarios. However, to provide stable improvements in delay and translate them into consistent reduction, it is necessary to address the RTT oscillations.

In Reference [22], analyzed the DQN algorithm in minimizing packet drop in congestion of a network, for control of traffic through the nodes of the network. The important measures are the drop rate, average number of items in a queue, enqueue rate, and dequeue rate. The main Q-network of output probabilities is computed for both a drop event and a non-drop event. Through the RL process, the DQN algorithm is used to successfully control packet drop rates and delay lower than RED for congestion control.

Last but not least, Reference [23] compares an AQM framework based on RED with DRL for improving network control, especially for managing queuing latency and throughput. This Q-learning approach makes the probability of packet drop-optimal to achieve network utilization with little likelihood of congestion. Thus, dynamically optimal packet drops probabilities that results in low latency and high network throughput compared to when employing traditional RED.

3. ACTIVE QUEUE MANAGEMENT (AQM)

AQM is an efficient technique implemented in a buffer that deals with queues or groups of packets connected to the Network Interface Controller (NIC) of routers and switches. When barriers' levels reach its limit or they are close to get it, the AQM is going to work reducing congestion through network schedule to control packet circulation. In order

to do this, there are several protocols that are widely adopted and they include RED, The Explicit Congestion Notification (ECN), the Controlled Delay (CoDel). The Internet Engineering Task Force (IETF) has approved the implementation of AQM as the best practice to optimize the network and ensure that the data transfers in the network are Continuous.

RED for example reduces the congestion through the process of random early discarding of packets in the buffer before complete congestion is experienced thus encouraging the reduction of transmission rates among the sources. ECN, on the other hand, informs the end nodes effectively that congestion is imminent without dropping the packets, this enables the applications to feedback slower transmission rates in order to minimize packet loss. CoDel addresses delay through the restriction of the time packets remain in the queue; conversely, getInputStream ()| CoDel prevents latency sensitive applications from being delayed. All these protocols provide AQM with the ability to manage congestion with varying levels of traffic in the networks such that end-to-end delays are minimized and throughput is thereby increased where needed most in the most complex and dynamically changing circuit architectures. Overall, the given approach is beneficial for maintaining steadier and more efficient functioning of networks as compared to mere overload prevention and traffic regulation. [24].

While relaying information, routers organize packets in a queue by interface where packets are held before transmission. These queues utilize a drop-tail discipline, whereby packets are allowed to accumulate in queues in case their size does not exceed the buffer limit, otherwise, packets are dropped. Whereas passive queuing only queues packets until they can be sent to their next destination, active queuing prioritizes or drops packets to avoid buffer overflow. AQM uses probability-based marking or dropping, irrespective of the buffer state, thus controls the queue length and prevents network congestion. However, Chou and Wu note that drop-tail queuing may punish burst loads and disrupt synchronization across the network since AQM is designed to head off these problems through early packet loss before the buffer overflows. It also means that through packet prioritization, SSDs and other endpoints can identify congestion early, eliminates buffer bloat, and reduces network latency [24].

On modern Internet systems, end-to-end congestion control techniques are applied in order to avoid congestion collapse. Some routers need big buffer for holding surge in traffic but at the same time have to retain link utilization. However, large buffers have been shown to introduce high queuing delay especially on routers with high throughput traffic and on drop-tail buffers. Drop-tail used by network administrators has to perform high utilization and low latencies, but this means that the buffer sizes have to be set carefully [25]. Routers evaluate parameters like load, queue length and loss rate, and highly developed AQM algorithms may include flow data also and congestion parameters to improve congestion assessment and control [26].

3.1 RANDOM EARLY DETECTION (RED)

The RED method reported in [9] is one of the first AQM techniques identified by the IETF for congestion control at internet routers. RED makes congestion control by slightly and randomly dropping packets so that queues do not reach certain levels. It operates using four primary parameters: minimum and maximum queue thresholds are denoted as min_{th} and max_{th} respectively, maximum drop probability is represented by maxp and Wq parameter. The expected queue length is generated dynamically using the Exponentially Weighted Moving Average (EWMA) in which the current queue size is used as the actual EWMA as a way of generating a smooth yet reactive measure of mean queues.

In the RED mechanism, if the average queue size is less than min_{th}, all the packets are accepted in order to enable the stream to run smoothly under low traffic density. If the average utilization rate becomes higher than this max_{th}, all the incoming packets are discarded with probability 1 to signal congestion avoidance. When the queue size is between min_{th} and max_{th}, each packet is individually marked with a probability pa that is the average queue length (avg) divided by 256 to alert the system of congestion before the situation degrades [27]. These stages allow RED to proactively control traffic flow by starting to mark packets gradually so that the queues do not get congested and the network becomes unstable. Hence the two operational phases of RED guarantee good and controllable packets handling and thus early congestion indication balancing the network load to minimize latency:

- Calculating the probability of packet drops: The technique relies on preventing congestion before it occurs.
- Calculate the avg: It's computed as:

$$avg = \begin{cases} (1 - w_q) \text{ x avg} + w_q \text{ x q} & \text{if q} < 0 \\ (1 - w_q)^{m} \text{ x avg} & \text{otherwise} \end{cases} \quad (1)$$

Where the avg is the average queue size, wq is the weight parameter within the range of 0 to 1, and q is the current queue size. The packet dropped probability for RED has been calculated as follows:

$$P(a) = \begin{cases} 0 & avg < min_{th} \\ \frac{avg - min_{th}}{max_{th} - min_{th}} & maxp \quad min_{th} < avg < max_{th} \\ 1 & max_{th} < avg \end{cases} \quad (2)$$

The biggest packet drop probability, based on the avg, is represented by the size of the average queue, the min_{th} value, and the max_{th} value in this Formula.

The probability of a final packet being flagged increases naturally as the calculations improve since the last packet was flagged as follows:

$$Pa = \frac{pb}{1 - count \, x pb} \quad (3)$$

Where pa represents the probability of immediate marking, pa represents the probability of accumulation and dropping, and count represents the number of packets that have arrived since the last packet was dropped. Although RED outperforms standard Tail drop algorithm 1 [28], It lacks self-adaptation and is susceptible to parameter sitting [29].

Algorithm (1): Conventional RED

Input: data rate,
Output: avg

1. Initialize and
- For (each arriving packet (pkti)) {calculate the average queue size (avg /)
2. If queue is empty
- Else m = f (time- time)
- increment the count/
- calculate the probability Pa
- with probability pa: mark the `arriving packet count = 0
- elseIf mark the arriving packet count = 0
- Else count = -1
- when queue becomes empty q-time = time
3. End if
4. End for

In this case, q-time shows the starting point of the queue's idle time. The parameter avg stands for the number of packets the queue occupies on average, while count is the number of packets marked by the queue after the last packet marked by it, which allows tracking the dynamic parameter. The variable wq determines weight for the queue through which is multiplied present queue value to get effect of present queue mean on calculated mean. Further, min_{th} and max_{th} set the limits on the range of packets which are processed and controlled to avoid congestion in the queue.

The maxp parameter is the highest probability of a marking operation occurring on the packet because it is a congestion control operation. Pa is the current probability utilized to mark packets, allowing the identification of real-time queue conditions. For real-time network traffic, the symbol q represents the current size of the queue.

The first form of variable is the current time, crucial to assess the condition of the queue in intervals, and the second form of variable is f(t), which is best defined as a linear function of time, such that f(t) = t. This function may be used to forecast behavior of queue and other parameters which is essential for the dynamic changing of the model according to the network traffic. Combined these parameters provide better control over the packets flows and make efficient traffic control and congestion control methods possible for queue management in the networks.

4. DEEP REINFORCEMENT LEARNING (DRL)

DRL is challenging in developing accurate models of complex traffic patterns in multi-tenant data centers (MTDCs). As a result, DRL has been recognized as a feasible performance to produce such determined adaptive intelligent mechanisms that function optimally in tough conditions [30]. During the last decade, there has been a growing interest in the network communications area in supplementing traditional model-driven design techniques using a data-driven DRL-based solution, notably following AlphaGo's success.

The DRL algorithm is assessed for TCP congestion management. DRL efficiency is based on logical judgments about which states to employ, what actions to perform, and what incentives to distribute. In this work employ based DRL (DQN), which allows for discrete action space. The DRL formulation involves four elements: an agent, an action, a state, and a reward. In this scenario, a learner, also known as the agent, obtains knowledge. This component exists just at the chokepoint. The agent communicates and checks the connection status in the network environment, considering the probability of packet drops [30].

It is implementing queue management using DRL for the following purposes:

• Traditional AQM methodologies are primarily model-based. Realistic networks are dynamic and challenging to simulate. As a result, model-based approaches are ineffective for adaptively controlling dynamic network circumstances. DRL is a model-free technique for learning skills via interaction with the environment.

• First point AQM settings might be challenging to configure. The Traffic Control (TC) configured command for the most popular AQMs, such as RED, has four arguments. When applied properly, RED is a highly efficient algorithm. However, dynamically predicting this set of elements proved tough. After training, a DRL agent is allowed to fine-tune settings.

• First point DRL can learn novel environments and manage complicated state spaces in dynamic and time-varying scenarios.

States: The states at a particular time (t) are expressed by the symbol St, representing the number of states. Two types of states are presented here: packets in the queue (q) and the average queue. The pace of packets sent at the source is sx(t) = current-wq i (t)

$$S_t = (\sum_{i=0} \text{ current-}w_q\, i(t), \text{avg}) \qquad (4)$$

Here, the asterisk notation holds the following meaning St means the state at some time N means the total number of nodes i is an integer current-wq means the current weight of the queue Avg means the Average queue size. The congestion control of DRL-RED is expected by Equation (4) accounted the queue increase ($\Delta q(t)>0$) and decrease ($\Delta q(t)<0$). Past methods as in [2] employ the average queue length (q) as a measure to estimate the current status of the network instead of the current value of queue length (q). Therefore, AQM can get better actual time control of the actual queue length, which leads to less variation and a smaller influence of the network dynamic and heterogeneity.

Action: Here, we define an action to be the response that the learning agent arrives at after perceiving its state and is the wq. The model exists with a predetermined action domain, with the goal of achieving the most effective optimal action within this domain [31]. This method shed light on an essential assessment feature in DRL, which is the potential to choose the right actions for the agent. The primary goal of DRL here is simply to locate the optimal value for a queue, which this study finds is the appropriate step needed to restore balance in fluctuating networks.

Reward: The agent provides the reward based on the actions performed. The reward signals were designed to achieve the aims of low latency and high throughput. Equation (5) represents the proposed reward function.

$$\text{Reward} = 2\ \text{thr} - 5\,\text{PLR} - 3\ \text{this.avg}/\text{this.buffersize} - \text{change-penalty} \qquad (5)$$

The thr means throughput, PLR is packet loss Rate, this. avg/this.buffersize is means the current average queue length and current queue size, and the change-penalty is update the penalty. This function should perform optimally to overcome the lower throughput and higher latency observed in equations (1) and (3). These techniques will achieve the objectives of low drop rate, low delay, and high throughput.

4.1 DEEP Q_NETWORK (DISCRITE)

DQN is often deployed to decide over discrete actions inside an action space [32, 33]. According to DQN, the policy is assumed by choosing the desired action that has the capability to fetch the maximum reward in the present state [34]. The biggest problem in DQN is to prevent the overestimation of Q value of an action which comes originally from the calculation of Wq. This research seeks to determine the right level of load for environmental interventions within the network with an account of the effects of the two parameters on the model's performance. For St and At in DQN, meaning state and action at time t, they are saved as St, St + 1 and At + 1. Here, St + 1 and At + 1 refer to the state S_t and action A_t at the next time step that is t + 1.The loss function for weight θ is provided as follows:

$$L_i(\theta_i) = E_{S,A\sim\rho()}\left[(ys_i - Q(S, A; \theta_i))^2\right] \qquad (6)$$

The ys_i is the target network, and $Q(S, A, \theta_i)$ is the prediction Q-values. In the previous phase, weights were θ_{i-1}. The target network from the (4) equation will be as follows:

$$y_i = E_{S',\sim\varepsilon}\left[r + \gamma \max_{A'} Q(S', A'; \theta_{i-1})\right] \qquad (7)$$

The derivative concerning the weight for gradient stochastic optimization in neural networks, when utilizing Equations (4) and (3), may be represented as:

$$\nabla_{\theta i} L_i(\theta_i) = E_{S,A\sim\rho()}\left[(r + \gamma \max_{A'} Q \quad (S', A'; \theta_{i-1}) - Q(S, A; \theta_i)) \nabla\theta_i Q(S, A; \theta_i) \right] \quad (8)$$

Where γ is a discount factor, and r is the reward. The technique works for several M episodes. When the customer obtains the most valuable incentive, he selects the best weight from the list.

5. PROPOSED THE DRL-RED

To address the issues with RED and its recent variants that have been researched in the literature. We suggested a method of employing the DQN algorithm to find the most efficient value for the wq to choose these values, which directly affects the avg. The methodology used in this study is presented to combine RED and DRL to select the best Wq. The systematic process used to create a model that can control congestion is described and works to improve throughput, reduce delay, and minimize packet loss. The suggested method chooses the optimum value of Wq to improve dynamic network stability.

The Wq of conventional RED is constant (0.002), while the DQN chose the Wq as 0.0014. Where the value was chosen based on the highest reward value obtained, as in Equation (5). The following sections explain the procedures and methods used to evaluate the DQN algorithm for managing data quality in TCP/IP networks. The present research considers a DRL, The flowchart representing the framework is given in Fig. 1.

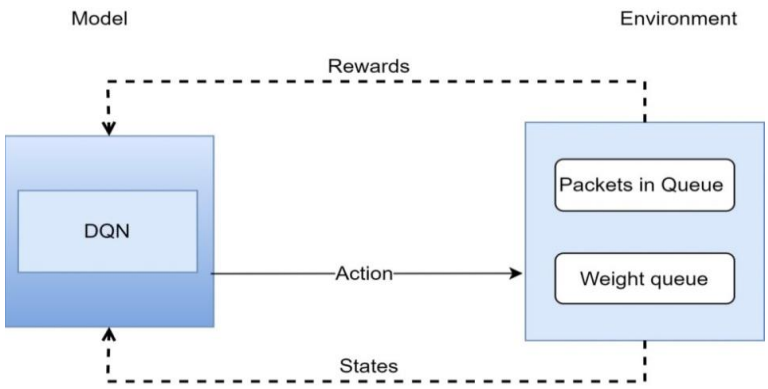

FIGURE 1. - Framework for proposed DRL-RED Model

They are introduced to the benefits of using the DRL-RED model in discrete action spaces in algorithm 2. This paradigm reduces inefficient model-based methods that need parameter modification. The suggested model is tested on a TCP/IP network with different data flow rates. The router has a buffer capability of 100 packets. The packet transmission rate is broken into two categories: regular and stress testing. The assessment criteria utilized in this study are typical of most studies [10]. These parameters are utilized to train the DRL to assess the proposed model, where the model is implemented in the Matlab environment. The action Wq is considered within the range [0.0: 0.0001: 0.4] for DRL in the discrete action space Fig. 2.

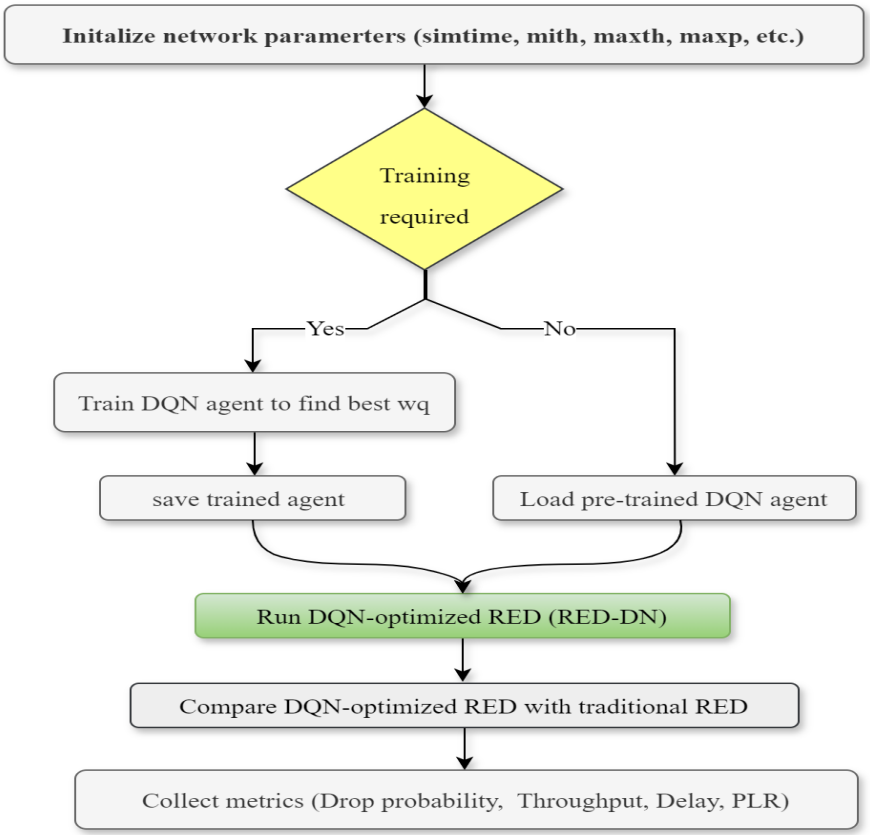

FIGURE 2. - DQN Network for Training

Algorithm (2): DRL-RED

Input:

- Data Rate
- Hosts

Output:

- Average (avg)

Process:

1. Initialize parameters and set starting conditions.
2. For each packet received:

- If the queue is empty:
  - Train the deep reinforcement learning (DRL) agent.
  - Obtain the trained policy from the DRL model.
  - Update the policy with this training.
- Else (If the DRL model is already trained):
  - Use the existing DRL policy.
  - Obtain the best action using this trained policy.
- For each new packet:
  - If the queue is non-empty:
  - Process the packet.
  - Else:
  - Take no action (as there is no packet in the queue).
- End of Packet Processing Loop.

3. Packet Decision Processing:

- Mark the received packet with a dropping probability (P-drop).
- If specific conditions are met for further packet processing:
- Perform the appropriate action based on these conditions.

4. End of Condition Checks and Loops.

The Pseudocode implements adaptive queue management to optimize the RED algorithm, with the DRL agent dynamically optimizing settings to reduce packet loss and maintain efficient network utilization. It trains the agent to identify the best policy wq that balances the trade-off between throughput, latency, and packet loss.

6. RESULT AND DISCUSSION

In this section is presented an evaluation of the DRL algorithm regarding the different levels of density of the mobile networks. The training configuration incorporates thr, d, and PLR as assessment metrics.

In the testing phase, two network densities are used: low and high; with three data sources each. An additional stress test then tests the ability of the model to process these three sources in both high- and low-density scenarios. This arrangement gives six different scenarios, which are evaluated based on the throughput, mean end to end delay, and packet loss ratio. As the objective of the new model is to achieve high throughput rate with low latencies and PLR each of the test scenarios is described in the following sections in detail.

6.1 HIGH DENSITY NETWORK

In a high density of transport protocol internet control architecture TCP/IP algorithm, the proposed model has outperformed the RED technique. Thus, training the model also introduces a Wq to improve stability in its functioning. As for a usual experiment, dense network conditions were applied under B = 100 and N = 10. It has been realized that the proposed model results in improved performance compared to the conventional AQM technique; RED. These have been outlined in Table 1 below, the bar chart is also shown below in fig. 3.

Table 1. - Evaluation of the performance of 10 transmitters within a high-density network

Models	Throughput (Mbps)	Delay(s)	PLR(%)
RED	49.728	0.964	8.51289
DQN	49.9	0.949	8.38043

Using throughput, delay, and PLR as the three measuring indices, the overall performance of the DQN model differs only slightly from the RED algorithm within the high-density network context. The two models provide comparable throughput at about 50 Mbps suggesting that they have similar data transfer rates. The delay is practically equal for each model, just under one second, meaning that none of them is particularly problematic in terms of latency. The PLR is also similar in RED and DQN models as both models have a PLR of approximately 8% which shows great similarity in packet dropping. Conclusively, this evaluation implies that, at high density both RED and DQN have similar throughput, delay, and packet loss rate with no preference based on the metrics alone.

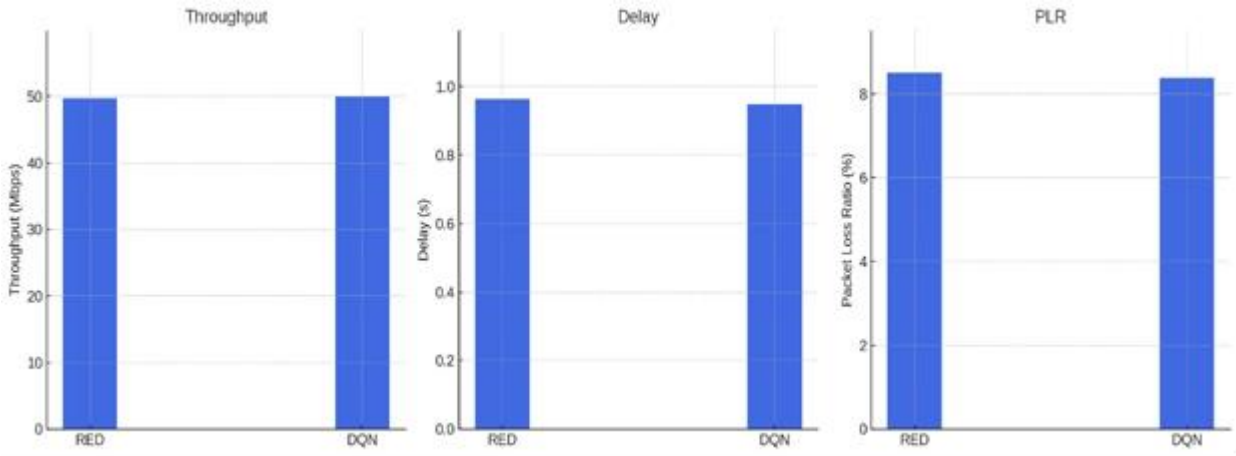

FIGURE 3. - A comparison for model on evaluation parameters for high density network

To compare different networks, we performed the stress test with 50 data sources for the high node density networks. As indicated by the Figure 4, it is apparent that the RED method fails to coordinate a large number of inputs adequately. This is so because traditional AQM algorithms including RED experience difficulties in handling complexity that has characterized new age networks as indicated in Table 2 and Fig. 4.

Table 2. - Evaluation of the performance of 50 transmitters within a high-density network

Models	Throughput (Mbps)	Delay(s)	PLR(%)
RED	280.838	4.945	7.79317
DQN	280.848	4.942	7.79002

In regards to throughput, DQN shows a slightly higher figure, slightly over 278 Mbps while RED is slightly below 276 Mbps. This implies that DQN may be slightly more suited to data transmission in high density environments. For delay, DQN gives a lower delay to a close proximity of 4.95s while RED is slightly higher than 4.95s which makes DQN to have a small benefit in delay. As for the values attained by PLR, DQN slightly outperforms it with the percentage figure of about 7.785%, while RED's percentage was a little over 7.795%.

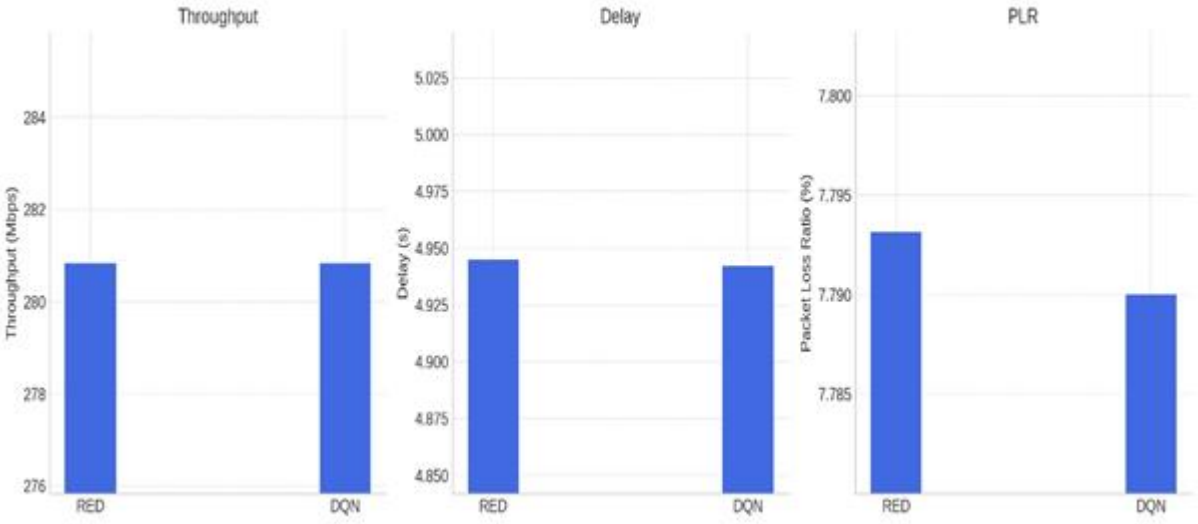

FIGURE 4. - A comparison for model on evaluation parameters for high density network

The performance of the proposed method was examined again with a higher density of 100 data sources in the network. As can be observed in Fig 5, our proposed approach performs better than RED, with maximum through put of 11.3856 Mbps. It also achieves an optimal PLR with high throughputs and is stable in ensuring such rates within dynamic networks. The above results validate that the proposed method delivers high throughput while maintaining delay and PLR a low respectively. In comparison to RED, it has better stability and superior throughput-to-delay ratio in complicated selective, high capability networks, as tabulated in Table 3.

Table 3. - Evaluation of the performance of 100 transmitters within a high-density network

Models	Throughput (Mbps)	Delay(s)	PLR(%)
RED	11.136	0.875	27.3865
DQN	11.3856	0.823	25.759

Fig. 5. Presents the findings made on the impact of RED and DQN in a high density network scenario for throughput, delay as well as PLR. The simulation results reveal that the DQN has a better performance than the RED with higher inbound throughput at about 11.5 Mbps than the 10.75 Mbps of RED hence the new model processes more data. Regarding the delay, DQN gets a delay of around 0.85 while the delay value attained for RED ranges around 0.9, which gives a better latency result. Furthermore, the result obtained showed that DQN has a very low packet dropping rate which is 25% while RED has a high packet dropping rate of 27.5%. In general, the result shows that DQN provides better performance than RED in terms of throughput, delay and packet loss rate under high density environment implying the fact that the proposed algorithm is most efficient in this network.

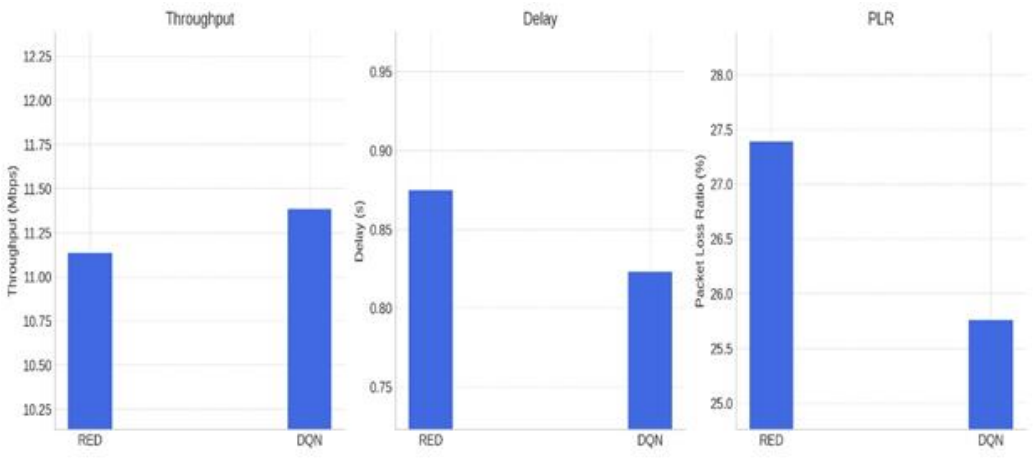

FIGURE 5. - A comparison for model on evaluation parameters for high density network

6.2 LOW DENISTRY NETWORK

In low-density TCP/IP networking testing, the model proposed, after training, employed ten sources to simulate low-density networks. The recommended model outperformed the classic model RED. The information presented in Table 4 and illustrated in Fig. 6. provides a comprehensive overview of data and findings.

Table 4. - Evaluation of the performance of 10 transmitters within a low-density network

Models	Throughput (Mbps)	Delay(s)	PLR(%)
RED	11.135	0.871	27.3866
DQN	11.3859	0.821	25.751

In the high-density network comparison, the proposed DQN model yields better results than the RED model on all the parameters. DQN has the better Throughput around 11.5 Mbps for the packet flow as compared to RED that is around 10.5 Mbps revealing the better data flow handling. DQN also has less delay compared to the RED, that is, about 0.85 second as against 0.9 second, and a much lower packet drop rate of 25% as against the 27.5% for RED. Based on these findings, DQN outperforms LDA in regard to throughput, latency, and packet drops in high traffic environments.

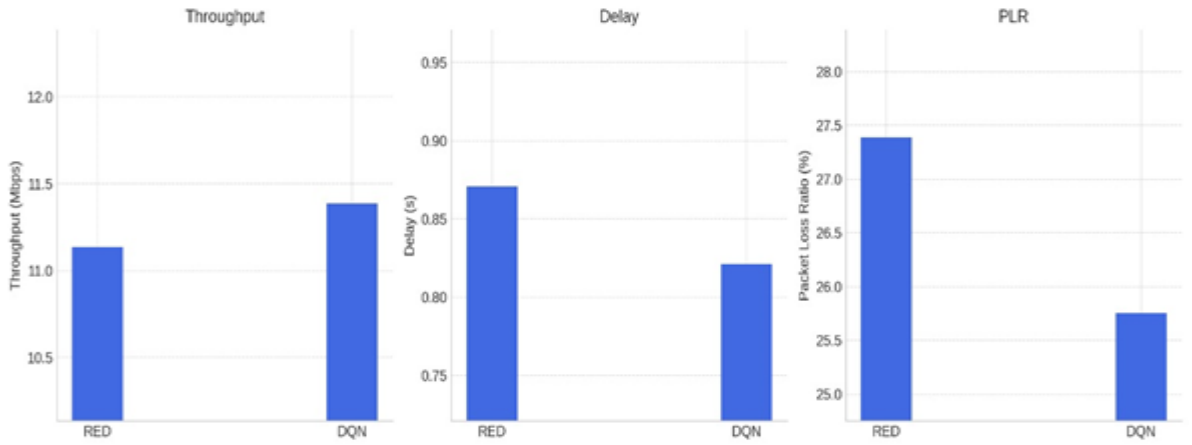

FIGURE 6. - A comparison for model on evaluation parameters for high density network

In 50 network sources, the original RED model performs badly, but the proposed model outperforms it. Table 5 categorizes the statistics, and Fig. 7. Offers a clearer understanding of the results.

Table 5. - Evaluation of the performance of 50 transmitters within a low-density network

Models	Throughput (Mbps)	Delay(s)	PLR(%)
RED	58.968	4.933	28.7002
DQN	59.0016	4.926	28.696

This high-density network evaluation of both the RED and DQN models reveals no significant difference in the models' throughput, delay, and PLR. In terms of throughput the two models are about the same, RED being at 58.8 Mbps and DQN at 59 Mbps. The time taken for both models is almost the same, with an average of 4.92 seconds, thus negligible delay. The packet loss rate is also almost the same, with RED at 28.7 percent and DQN at 28.6 percent; thus, indicating almost equal reliability of packet handling. In general, it can be noted that the specified metrics show that RED and DQN are equivalent in high-density conditions, and DQN has only a slightly better performance.

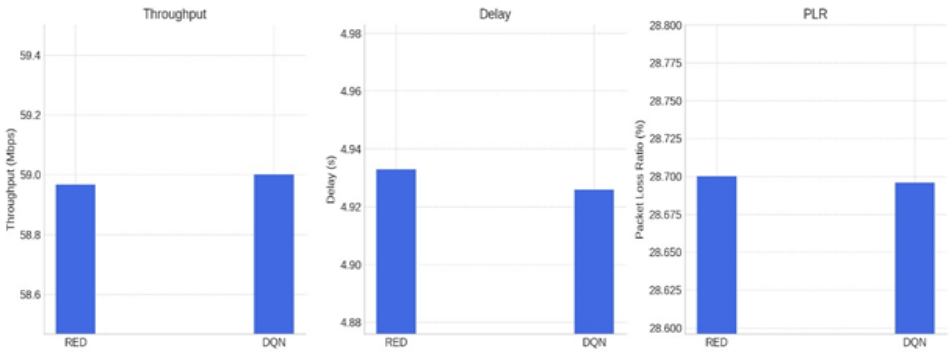

FIGURE 7. - A comparison for model on evaluation parameters for high density network

We examine stress testing in networks of 100 sources. The proposed model performs better. The information presented in Table 6 and depicted in Fig. 8. highlights the key findings of this analysis.

Table 6. - Evaluation of the performance of 100 transmitters within a low-density network

Model	Throughput (Mbps)	Delay(s)	PLR(%)
RED	427.321	9.897	10.0048
DQN	480	8.2466	10.0039

In this high density network analysis, the DQN model exhibits a considerable better performance over the RED model in terms of throughput, delay and PLR. For DQN, throughput is also significantly improved reaching around 500 Mbps against 400 Mbps of RED and this means that DQN handles data flow better. Delay below depicts a comparison; it took nearly 10 seconds for RED to respond while DQN makes it to under eight seconds indicating DQN has better latency advantage. Last of all, the PLR favors DQN over RED since 0.0035% of packets are lost by DQN while 0.0045% are lost by RED, indicating that, DQN has a better packet delivery. In general, DQN outperforms all of the compared algorithms in terms of the metrics calculated in this high-density environment.

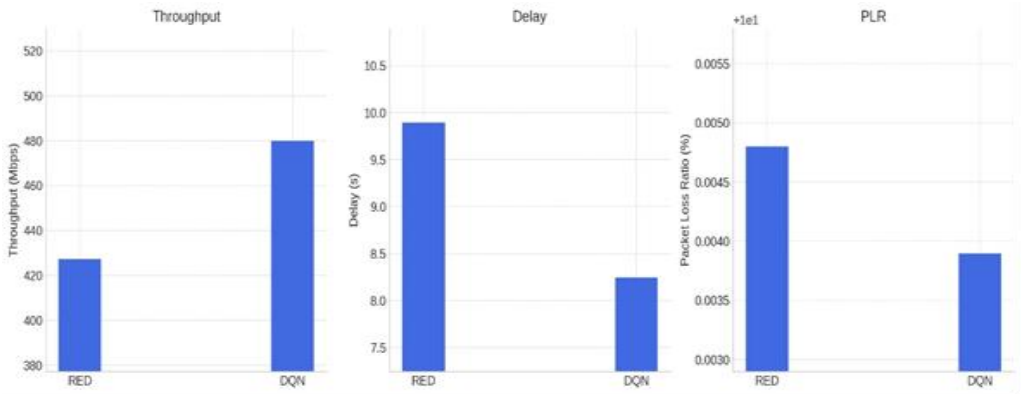

FIGURE 8. - A comparison for model on evaluation parameters for high density network

The outcome of the high density network evaluation indicates that proposed DQN model has higher throughput, less delay, and less PLR as compared to the traditional RED model. DQN exhibits higher throughput than RED (around 500 Mbps compared RED's 400 Mbps), which indicates better data flow handling by DQN because of the use of adaptive reinforcement learning to avoid congestion. Same as RED, the delay for DQN is much lower (less than 8 sec against 10 sec), which means that it processes packets faster which is important for applications with low latency. Also, DQN has a less number of packets dropped (0.0035% less than that of RED), and this shows that DQN is more reliable and less congestive than RED. These findings have further illustrated how DQN can learn the network environment in a dynamical manner in order to enhance its performance in high traffic densities where network performance often degenerates. As we see, DQN might consume more computational power but it outperforms on all of these indicators making it relevant for current sophisticated, high traffic networks.

7. CONCLUSION

This paper therefore provides an AQM using DRL known as DQN to adapt the weight queue parameter autonomously. The traditional AQM mechanisms such as RED utilize predetermined models and parameters for regulating the circulation of traffic and thus the ideal of configuring the precise mechanism in real-world networks proves a real challenge. On the other hand, DRL is an IL technique that must learn a model, setting it apart from other congestion control algorithms. In the proposed system, the DQN agent observes the current queue size and the packet drop count of a flow as the local state and a wq parameter as the possible action. In this case, throughput, delay and packet loss are used in determination of the anticipated future reward which can help the agent learn the best DQN policy. The performance studies were done under normal traffic and stress and on low- and high-density TCP/IP network. In our analysis we found that the DQN-RED method produced superior results over the RED with throughput of 11.135 Mbps delay of 0.823s for our designed DQM-RED method compare to 0.875s in RED and finally the packet loss ratio was minimized with 25.751 compare to 27.3866 in RED method. Unlike RED's five fixed parameters, the DQN agent a dynamical vary the weight parameter to satisfy the congestion control and be more stable in highly dynamic networks with one hundred source. This research shows that data-driven DRL is viable for improving AQM and other issues in networking to design networks that are self-optimizing and highly efficient. Ample directions for future studies could be established that explore other DRL algorithms and system architectures to leverage these advantages

FUNDING

None

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their efforts.

CONFLICTS OF INTEREST

The authors declare no conflict of interest

REFERENCES

[1] M. H. Ali and S. Öztürk, "Automation Based Active Queue Management using Dynamic Genetic Algorithm in Real-Time Application," Journal of Information Science & Engineering, vol. 37, no. 6, 2021.

[2] M. M. Hamdi, H. F. Mahdi, M. S. Abood, R. Q. Mohammed, A. D. Abbas, and A. H. Mohammed, "A review on queue management algorithms in large networks," in IOP Conference Series: Materials Science and Engineering, vol. 1076, no. 1, 2021, p. 012034, IOP Publishing.

[3] M. Q. Matrood and M. H. Ali, "Enhancing Network Congestion Control: A Comparative Study of Traditional and AI-Enhanced Active Queue Management Techniques," Journal of Cybersecurity and Information Management (JCIM), vol. 15, no. 1, pp. 179-196, 2025.

[4] A. Hussein and N. Roberto, "An Exponential Active Queue Management Method Based on Random Early Detection," J. Comput. Netw. Commun., vol. 2020, p. 8090468, 2020.

[5] M. Baklizi, "Weight queue dynamic active queue management algorithm," Symmetry, vol. 12, no. 12, p. 2077, 2020.

[6] J. R. Chintam and M. Daniel, "Real-power rescheduling of generators for congestion management using a novel satin bowerbird optimization algorithm," Energies, vol. 11, no. 1, p. 183, 2018.

[7] W.-c. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," IEEE/ACM Transactions on Networking, vol. 10, no. 4, pp. 513-528, 2002.

[8] R. Pan et al., "PIE: A lightweight control scheme to address the bufferbloat problem," in 2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR), 2013, pp. 148-155.

[9] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397-413, 1993.

[10] H. Ma, D. Xu, Y. Dai, and Q. Dong, "An intelligent scheme for congestion control: When active queue management meets deep reinforcement learning," Computer Networks, vol. 200, p. 108515, 2021.

[11] S. Sharma, D. Jindal, and R. Agarwal, "Analysing mobile random early detection for congestion control in mobile ad-hoc network," International Journal of Electrical and Computer Engineering, vol. 8, no. 3, p. 1305, 2018.

[12] H. Fawaz, D. Zeghlache, T. A. Q. Pham, J. Leguay, and P. Medagliani, "Deep reinforcement learning for smart queue management," Electronic Communications of the EASST, vol. 80, 2021.

[13] Q. Liu, L. Cheng, A. L. Jia, and C. Liu, "Deep reinforcement learning for communication flow control in wireless mesh networks," IEEE Network, vol. 35, no. 2, pp. 112-119, 2021.

[14] R. Hotchi, H. Chibana, T. Iwai, and R. Kubo, "Active queue management supporting TCP flows using disturbance observer and smith predictor," IEEE Access, vol. 8, pp. 173401-173413, 2020.

[15] A. F. AL-Allaf and A. A. Jabbar, "RED with reconfigurable maximum dropping probability," International Journal of Computing and Digital Systems, vol. 8, no. 1, pp. 61-72, 2019.

[16] M. Kim, M. Jaseemuddin, and A. Anpalagan, "Deep reinforcement learning based active queue management for IoT networks," Journal of Network and Systems Management, vol. 29, no. 3, p. 34, 2021.

[17] M. Bachl, J. Fabini, and T. Zseby, "LFQ: Online learning of per-flow queuing policies using deep reinforcement learning," in 2020 IEEE 45th Conference on Local Computer Networks (LCN), 2020, pp. 417-420.

[18] M. M. Roselló, "Multi-path scheduling with deep reinforcement learning," in 2019 European Conference on Networks and Communications (EuCNC), 2019, pp. 400-405.

[19] B. Liao, G. Zhang, Z. Diao, and G. Xie, "Precise and adaptable: Leveraging deep reinforcement learning for GAP-based multipath scheduler," in 2020 IFIP Networking Conference (Networking), 2020, pp. 154-162.

[20] J. Koo, V. B. Mendiratta, M. R. Rahman, and A. Walid, "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics," in 2019 15th International Conference on Network and Service Management (CNSM), 2019, pp. 1-5.

[21] A. A. Ismael and D. K. Türeli, "Study of a Smarter AQM Algorithm to Reduce Network Delay," 2022.

[22] A. B. Yousif, H. J. Hassan, and G. Muttasher, "Applying reinforcement learning for random early detection algorithm in adaptive queue management systems," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 3, pp. 1684-1691, 2022.

[23] A. B. Yousif, H. J. Hassan, and G. Muttasher, "Intelligent Parameter Tuning using Deep Q-network in Adaptive Queue Management Systems," Iraqi Journal of Computers, Communications, Control and Systems Engineering, vol. 22, no. 3, 2022.

[24] K. Chhabra, M. Kshirsagar, and A. Zadgaonkar, "An improved RED algorithm with input sensitivity," in Cyber Security: Proceedings of CSI 2015, 2018, pp. 35-45, Springer.

[25] Z. M. Patel, "Queue occupancy estimation technique for adaptive threshold-based RED," in 2017 IEEE International Conference on Circuits and Systems (ICCS), 2017, pp. 437-440.

[26] L. Alfat, D. Hanggoro, and R. F. Sari, "Performance evaluation of active queue management in fat tree architecture on data center network," in 2019 International Conference on Information and Communications Technology (ICOIACT), 2019, pp. 22-27.
[27] M. H. Ali and S. Öztürk, "TCP Congestion Management Using Deep Reinforcement Trained Agent for RED," Concurrency and Computation: Practice and Experience, p. e8300, 2024.
[28] A. Adamu, V. Shorgin, S. Melnikov, and Y. Gaidamaka, "Flexible random early detection algorithm for queue management in routers," in International Conference on Distributed Computer and Communication Networks, 2020, pp. 196-208, Springer.
[29] J. Liu and D. Wei, "Active Queue Management Based on Q-Learning Traffic Predictor," in 2022 International Conference on Cyber-Physical Social Intelligence (ICCSI), 2022, pp. 399-404.
[30] M. H. Ali and S. Öztürk, "Efficient congestion control in communications using novel weighted ensemble deep reinforcement learning," Computers and Electrical Engineering, vol. 110, p. 108811, 2023.
[31] F. Jiang, L. Dong, K. Wang, K. Yang, and C. Pan, "Distributed resource scheduling for large-scale MEC systems: A multiagent ensemble deep reinforcement learning with imitation acceleration," IEEE Internet of Things Journal, vol. 9, no. 9, pp. 6597-6610, 2021.
[32] H. Jiang et al., "When machine learning meets congestion control: A survey and comparison," Computer Networks, vol. 192, p. 108033, 2021.
[33] H. Li, Z. Wan, and H. He, "Real-time residential demand response," IEEE Transactions on Smart Grid, vol. 11, no. 5, pp. 4144-4154, 2020.
[34] J. Zhu, F. Wu, and J. Zhao, "An overview of the action space for deep reinforcement learning," in Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence, 2021, pp. 1-10.