

Article

Lightweight Adaptive Reinforcement Learning-Based TCP Congestion Control for Multi-Hop Ad Hoc Networks

Hai Li ^{1,2,*}  and Zhe Xin ¹ 

¹ School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China; 3120230676@bit.edu.cn

² THz Science and Application Research Center, Beijing Institute of Technology, Zhuhai 519088, China

* Correspondence: haili@bit.edu.cn

Abstract

Ad hoc networks are characterized by flexible deployment and multi-hop communication, which has facilitated their growing prevalence in diverse applications. However, the TCP protocol exhibits substantial performance degradation in multi-hop ad hoc networks with dynamic topologies. To address this issue, this paper proposes TCP-RLA, a lightweight adaptive reinforcement learning-based TCP congestion control algorithm. It predicts network state variations and leverages a deep Q-network (DQN) with a rule-assisted discrete action space to adaptively tune the congestion window. This design boosts convergence speed and reduces computational complexity, making it well-suited for resource-constrained ad hoc nodes. Simulation results demonstrate that, compared with two reinforcement learning-based algorithms (GVegas and Orca), TCP-RLA achieves an average throughput improvement of 36.1% and 43.3%, an average round-trip time (RTT) reduction of 13.1% and 47.9%, and an average packet loss rate (PLR) reduction of 33.3% and 50%, respectively.

Keywords: ad hoc; TCP congestion control; reinforcement learning

1. Introduction

In recent years, ad hoc networks have attracted significant research interest due to their self-healing capabilities, scalability, decentralization, and diverse applications [1]. The TCP protocol was originally designed for wired network environments, and its congestion control mechanism performs well in wired networks [2]. However, direct deployment of the TCP protocol in ad hoc networks yields suboptimal performance, owing to their dynamic topology, shared medium, and error-prone nature [3]. Classical TCP congestion control schemes leverage specific signals to identify congestion and then adjust the congestion window (CWND) or transmission rate per predetermined rules; these are called rule-based schemes. Rule-based TCP congestion control mechanisms can be categorized into loss-based mechanisms and delay-based mechanisms. These methods suffer from limitations such as difficulty in adapting to new networks, lack of utilization of historical information, and performance degradation due to limited human understanding of the network [4]. Rule-based TCP congestion control mechanisms are no longer applicable to multi-hop ad hoc networks, resulting in reduced communication efficiency and stability. Given the characteristics of multi-hop ad hoc networks, improving the TCP congestion control mechanism enables more stable and efficient communications.



Academic Editor: Dimitris Kanellopoulos

Received: 28 January 2026

Revised: 20 February 2026

Accepted: 23 February 2026

Published: 25 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Facing complex and ever-changing network scenarios, congestion control algorithms combined with machine learning are more versatile [5]. Compared with traditional rule-based TCP congestion control algorithms, machine learning-based TCP congestion control algorithms can utilize historical information and adaptively adjust to the network environment. Machine learning-based congestion control algorithms can be divided into those based on supervised and unsupervised learning and those based on reinforcement learning (RL). Reinforcement learning is more effective in handling dynamic and stochastic real-world network congestion because it has higher online learning capabilities [6].

To the best of our knowledge, existing RL-based congestion control algorithms suffer from limitations such as restricted application scenarios, high computational complexity, dependence on underlying information, and a trade-off between efficiency and packet loss [7]. Therefore, we propose TCP-RLA, a lightweight adaptive reinforcement learning TCP congestion control algorithm, suitable for multi-hop ad hoc networks. This algorithm has the following characteristics:

1. A long short-term memory (LSTM) network is employed to predict next-time-step network parameters, enabling the reinforcement learning algorithm to make more informed decisions regarding sudden network changes.
2. Given the limited computing resources of ad hoc network nodes, the RL model DQN is adopted—featuring lower computational complexity than proximal policy optimization (PPO) and deep deterministic policy gradient (DDPG) [8]. Meanwhile, the discrete action space integrated with a rule-based algorithm enables effective congestion window adjustment while reducing computational overhead and accelerating convergence.
3. The proposed scheme acts as a drop-in replacement for standard TCP on unmodified end hosts, requiring no lower-layer network parameters. It only uses transport-layer information natively available to TCP senders and coexists harmoniously with other TCP variants in ad hoc networks.
4. A comprehensive optimization approach is adopted to enhance network performance by incorporating throughput, RTT, and PLR metrics.

The remainder of this paper is organized as follows. Section 2 introduces related work on the TCP congestion control mechanism based on machine learning. Section 3 states the proposed TCP-RLA scheme. The simulation results are presented in Section 4. The analysis and discussion of the simulation results are given in Section 5. Finally, the conclusion is presented in Section 6.

2. Related Works

In recent years, many studies have applied machine learning algorithms to TCP congestion control. For instance, Yan et al. [9] propose the Indigo algorithm, which stores the mapping relationship between network states and control actions in a long short-term memory (LSTM) network and periodically adjusts the congestion window according to real-time network state feedback. Molia et al. [10] develop TCP-RLLD, a deep learning-based scheme that predicts the root causes of packet loss to avoid unnecessary transmission rate reductions triggered by non-congestion losses. B. Pillai et al. [11] put forward TCP-CNNLSTM, which collects network traffic information, trains a hybrid model combining convolutional neural networks (CNNs) and LSTM, and ultimately classifies network states to select the optimal congestion management strategy. However, these machine learning-driven TCP congestion control algorithms, which are predicated on network state classification, are only tailored to specific application scenarios and thus cannot accommodate the highly dynamic and heterogeneous characteristics of multi-hop ad hoc networks.

Against this backdrop, researchers have shifted their focus to reinforcement learning (RL) techniques, as RL-enabled TCP congestion control algorithms can implement adaptive network regulation and thereby achieve superior performance across diverse complex environments. Majid Hamid Ali et al. [12] propose a weighted ensemble deep reinforcement learning (DRL) model to solve the TCP congestion control problem. It combines four DRL models to achieve efficient congestion control in active queue management (AQM) systems. Jay et al. [13] present Aurora, which constructs the input state space using the concatenation of network states from the previous 10 time steps to capture more comprehensive historical information for decision-making. Mahajan et al. [14] develop TCP-Int, an RL-based congestion control algorithm implemented on the ns3-gym framework. Despite their enhanced adaptability, these RL/DRL-enabled algorithms still exhibit limited scenario generalization.

To further enhance the performance of RL-based congestion control schemes, a subset of studies has explored the integration of network state prediction mechanisms into algorithm design. Jiang et al. [15] propose TCP-Gvegas, an improved TCP Vegas variant based on gray prediction theory tailored for multi-hop ad hoc networks. This scheme incorporates lower-layer network parameters into the theoretical throughput model and utilizes gray prediction-based future throughput instead of actual measured throughput to facilitate real-time online control; it also employs Q-learning to determine a more rational congestion window adjustment step size. While this prediction-augmented approach enhances the responsiveness of online control, it still suffers from critical drawbacks, such as high computational complexity and heavy reliance on fine-grained underlying network information.

Recognizing the inherent trade-off between the long-proven stability of rule-based schemes and the adaptive advantage of data-driven DRL methods, recent research has turned to hybrid frameworks that integrate both paradigms. Typical hybrid strategies include using rule-based algorithms to accelerate the training process of DRL agents or leveraging DRL to optimize the decision-making logic of rule-based algorithms. For example, Abbasloo et al. [16] propose a hybrid algorithm combining Cubic and DRL to mitigate the slow convergence speed and high computational complexity of pure DRL schemes. Alipio et al. [17] design a cache-aware TCP congestion control algorithm, which adopts a rule-based exponential growth strategy during the fast start phase and switches to a DRL-driven decision mechanism to select one of three window adjustment strategies when the network approaches a congested state. Li et al. [18] integrate a BBR-like bandwidth detection mechanism into RL-based congestion control, helping the algorithm adapt to high-speed dynamic bandwidth network environments and improve bandwidth utilization. Nevertheless, these hybrid algorithms face a nontrivial trade-off between transmission efficiency and packet loss rate, failing to strike a full balance of stability, adaptability, and lightweight deployment in multi-hop ad hoc networks.

Table 1 summarizes the shortcomings of the TCP congestion control algorithms discussed in the aforementioned literature.

Table 1. Comparative summary of TCP congestion control algorithm.

Proposal	Approach	Key Features	Main Limitations
Indigo [9]	LSTM	Uses LSTM to map network states to actions	Only applicable to specific network scenarios
TCP-RLLD [10]	Loss differentiation, RL	Uses RL to predict cause of a packet loss	Cannot adapt to dynamic topology changes in multi-hop ad hoc networks

Table 1. Cont.

Proposal	Approach	Key Features	Main Limitations
TCP-CNNLSTM [11]	CNN, LSTM	Classifies states and selects the optimal congestion control mode	Overly simple action space limits performance
Ensembled DRL AQM [12], Aurora [13], TCP-Int [14]	DRL	Uses DRL to adjust congestion window from historical experience	Limited generalization across different network scenarios
TCP-GVegas [15]	Grey prediction, Q-Learning	Predicts future throughput based on grey prediction for online control.	High computational complexity
Orca [16]	DRL, hybrid congestion control	Combines classic congestion control strategies and DRL	Not designed for dynamic multi-hop ad hoc environments
DRL-CaCC [17]	DRL, cache-aware congestion control	Avoids congestion while ensuring maximum cache utilization	Requires intermediate node caching information
RL-explore [18]	RL, bandwidth detection	A bandwidth detection mechanism is added to RL	Overly aggressive action space unsuitable for ad hoc networks

3. Methods

In this section, we present the overall architecture of the proposed congestion control algorithm. We also describe the functional implementation of its two core modules: a prediction module and a reinforcement learning module. The proposed algorithm is designed to achieve efficient and stable TCP transmission in dynamic multi-hop ad hoc networks.

3.1. Problem Formulation

Multi-hop ad hoc networks consist of mobile nodes connected by wireless links. These nodes can move freely and randomly, leading to highly dynamic network topologies. Such random topology changes make end-to-end routing difficult to predict at any time. This unpredictability further degrades network performance across different communication layers.

For congestion control problems in multi-hop ad hoc networks, the algorithm design must consider the following QoS metrics [19,20]:

1. Improve the throughput of the communication link;
2. Reduce end-to-end latency;
3. Lower the packet loss rate.

Accordingly, our algorithm aims to improve throughput while reducing latency and packet loss. Thus, this optimization problem is formulated as

$$\min(-\log(\text{throughput}) + \delta \frac{rtt}{rtt_{min}} + \gamma PLR) \quad (1)$$

In Equation (1), rtt is the current round-trip time, and rtt_{min} is the minimum rtt within a recent time window. PLR is the packet loss rate. Parameters δ and γ are weights that represent the relative importance of the respective metrics. A larger δ indicates a higher priority for low-latency communication. Similarly, a larger γ reflects greater sensitivity to packet loss in the network.

3.2. Proposal Approach

To address the above optimization goal, we model the congestion control problem in multi-hop ad hoc networks as a Markov decision process (MDP). A DQN-based reinforcement learning agent is adopted to make adaptive control decisions. The RL agent assesses the current network conditions based on the input state and the reward signal. It then selects the most appropriate action from a discrete action set to adjust the congestion window. The overall framework of TCP-RLA is shown in Figure 1.

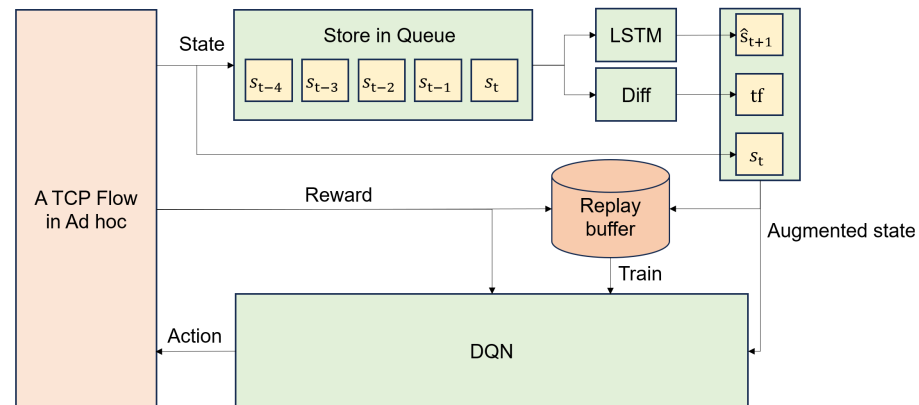


Figure 1. Overall framework of algorithm.

A key challenge in ad hoc networks is frequent and rapid network state variation. To handle this issue, we introduce an LSTM-based prediction module. This module estimates the next network state $\hat{s}_{t+1}$ based on historical states $\{s_{t-3}, s_{t-2}, s_{t-1}, s_t\}$. Meanwhile, we compute the first-order and second-order average rates of change from the historical states $\{s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}, s_t\}$. These statistics form a temporal feature tf . The current state s_t , temporal feature tf , and predicted state $\hat{s}_{t+1}$ are combined to form an augmented state for the DQN. By incorporating predicted future states and temporal features, the DQN can anticipate network variations in advance. This allows earlier and more accurate congestion window adjustments. Consequently, the agent can thus avoid congestion while fully utilizing available network bandwidth.

3.2.1. Prediction Module

The prediction module is designed to capture and forecast network state variations. It provides the RL module with enriched temporal information for more robust decision-making.

For the proposed algorithm, six network state parameters are collected: CWND, segment size, segments acked, bytes in flight, RTT, and minRTT. A buffer queue is used to store state values from the most recent five time steps. At each step of the RL process, the latest normalized state is appended to the queue. The prediction module operates as follows:

First, the LSTM network predicts the next network state $\hat{s}_{t+1}$. The input is the four recent states $\{s_{t-3}, s_{t-2}, s_{t-1}, s_t\}$. The output is the predicted state $\hat{s}_{t+1}$. During LSTM training, the input sequence uses the four earlier states $\{s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}\}$, and the label is the subsequent state s_t .

Second, we compute the first-order and second-order average rates of change of network states. These form the temporal feature tf , which helps the model capture dynamic trends.

The average first-order rate of change is defined as

$$tf_{fo} = \frac{1}{4} \sum_{i=-4}^4 \frac{s_{t+i+1} - s_{t+i}}{s_{t+i}} \quad (2)$$

The second-order differences are calculated as

$$d_i = s_{t+i+1} - s_{t+i}, (i = -4, -3, -2, -1) \quad (3)$$

The average second-order rate of change is then given by

$$tf_{so} = \frac{1}{3} \sum_{i=-4}^3 \frac{d_{i+1} - d_i}{d_i} \quad (4)$$

Third, s_t , tf , and $\hat{s}_{t+1}$ are concatenated and fed into the reinforcement learning module as the augmented input state.

The algorithm flow of the prediction module is shown in Algorithm 1.

Algorithm 1 Prediction Module

Train the LSTM network using collected network samples.

For each step, do

Obtain the current network state from the environment and update the queue buffer.

Extract the latest four states $\{s_{t-3}, s_{t-2}, s_{t-1}, s_t\}$ as LSTM input.

Run LSTM inference to obtain the predicted state $\hat{s}_{t+1}$.

Calculate the temporal feature tf according to Equations (2)–(4).

Feed s_t , tf and $\hat{s}_{t+1}$ to the RL module.

3.2.2. Reinforcement Learning Module

The reinforcement learning module is built upon the Deep Q-network (DQN). It uses the augmented state to achieve stable congestion control. DQN is based on the Bellman optimality equation, which can be given by

$$q(s, a) = E[R_{t+1} + \gamma \max_{a \in A(S_{t+1})} q(S_{t+1}, a) | S_t = s, A_t = a] \quad (5)$$

The action-value function is approximated by a deep neural network, denoted $\hat{q}(S, A, \omega)$, where ω is the neural network parameter, S represents the state space, and A represents the action space. The learning objective is to make $\hat{q}(S, A, \omega)$ approximate the Temporal Difference target (TD target). The loss function can be formulated by

$$J(\omega) = E[(R + \gamma \max_{a \in A(S')} \hat{q}(S', a, \omega) - \hat{q}(S, A, \omega))^2] \quad (6)$$

- Value network and target network

To reduce the complexity of calculating the gradient of the loss function in Equation (6), TD target y is treated as a constant. It is defined as

$$y = R + \gamma \max_{a \in A(S')} \hat{q}(S', a, \omega) \quad (7)$$

This formulation introduces two separate networks:

1. A value network $\hat{q}(S, A, \omega)$, whose parameters ω are updated at each step;
2. A target network $\hat{q}(S, A, \omega_T)$, whose parameters ω_T are fixed for a period T and then periodically updated from the value network;

By replacing the TD target in Equation (6) with the output of the target network $\hat{q}(S, A, \omega_T)$, the loss function can be rewritten as

$$J(\omega) = E[(R + \gamma \max_{a \in A(S')} \hat{q}(S', a, \omega_T) - \hat{q}(S, A, \omega))^2] \quad (8)$$

In this formulation, y remains constant within each period T when computing the gradient of $J(\omega)$. The gradient can thus be expressed as

$$\nabla_{\omega} J = E[(R + \gamma \max_{a \in A(S')} \hat{q}(S', a, \omega_T) - \hat{q}(S, A, \omega)) \nabla_{\omega} \hat{q}(S, A, \omega)] \quad (9)$$

- Experience replay

During training, samples are not used in the order they were collected. Instead, samples are stored in a replay buffer. In each iteration, a mini-batch of samples $\{(s, a, r, s')\}$ is taken from the replay buffer in a uniform distribution. The TD target y_T is calculated according to Equation (7), and the TD error is minimized by the mini-batch $\{(s, a, y_T)\}$, i.e., $J(\omega)$ in Equation (8).

In this work, the observation space $\mathcal{S}$ of the DQN is the augmented state, which can be given by

$$\mathcal{S} = [s_t, tf, \hat{s}_{t+1}] \quad (10)$$

The action space $\mathcal{A}$ includes five discrete CWND adjustment policies: no change, fast increase, slow increase, slow decrease, and fast decrease. They are defined as

$$\mathcal{A} = \{0; +\frac{1}{cwnd \times \beta}; +\frac{1}{cwnd \times \beta \times \beta}; -\frac{1}{cwnd}; -\frac{\beta}{cwnd}\} \quad (11)$$

To make adjustments more adaptive to congestion levels, we introduce a congestion coefficient β , which can be given by

$$\beta = \frac{rtt}{rtt_{min}} \quad (12)$$

This coefficient reflects the current congestion level by comparing the current rtt with the recent minimum rtt. Dividing the increment by β suppresses CWND growth under high congestion. Multiplying the decrement by β limits window reduction when the congestion level is reduced.

Regarding the setting of DQN rewards, we consider network metrics such as throughput, latency and PLR. The total reward consists of throughput reward, delay penalty, and loss penalty, which can be represented by

$$R = w_0 \times r_{thru} - w_1 \times p_{delay} - w_2 \times p_{loss} \quad (13)$$

The reward function in Equation (13) is derived from Equation (1) via heuristic reward shaping. Weights w_0, w_1, w_2 reflect the relative importance of each reward component.

Meanwhile, the throughput reward is defined as

$$r_{thru} = \log(\text{throughput}) \quad (14)$$

In Equation (14), the throughput is calculated based on the actual TCP-related packets to ensure its consistency with the real network environment.

In addition, the delay penalty can be expressed by

$$p_{delay} = \left(\frac{rtt}{rtt_{min}} - 1 \right) \quad (15)$$

The loss penalty can be given by

$$p_{loss} = PLR \quad (16)$$

The algorithm flow of the reinforcement learning module is shown in Algorithm 2.

Algorithm 2 Reinforcement Learning Module

Initial DQN parameter ω , policy π , small $\epsilon > 0$.

For each step, do

 Generate experience samples by π , and store them in a replay buffer $\mathcal{B} = \{(s, a, r, s')\}$.

 Draw a mini-batch of samples from $\mathcal{B}$ following a uniform distribution.

 For each sample (s, a, r, s') , calculate the target value $y_T = R + \gamma \max_{a \in A(S')} \hat{q}(S', a, \omega_T)$, ω_T

is the parameter of the target network.

 Update value network to minimize $J(\omega)$ in Equation (8) using batch $\{(s, a, y_T)\}$

 Set $\omega_T = \omega$ every C steps.

 Update policy:

$$\pi(a|s) = \begin{cases} 1 - \frac{\epsilon}{|A(s)|} (|A(s)| - 1), & \text{if } a = \underset{a \in A(s)}{\operatorname{argmax}} \hat{q}(s, a, \omega) \\ \frac{\epsilon}{|A(s)|}, & \text{otherwise.} \end{cases}$$

4. Simulation Results

This section presents the congestion control performance of the proposed TCP-RLA algorithm in multi-hop ad hoc networks. We compare TCP-RLA with two RL-based algorithms, namely GVegas [15] and Orca [16], as well as three traditional rule-based algorithms: LP [21], TCP Vegas, and TCP Cubic. Three standard performance metrics are evaluated: throughput, round-trip time (RTT), and packet loss rate (PLR) [22,23].

All simulation experiments are conducted on NS-3. We use ns3-gym [24] to enable interaction between the Python-based reinforcement learning module and the NS-3 network environment. The ns3-gym framework integrates OpenAI Gym with NS-3 to facilitate RL-based research in networking. All implementations are based on Python 3.8.10 and OpenAI Gym 0.26.2. The main simulation parameters are summarized in Table 2.

Table 2. NS-3 simulation parameters.

Parameters	Value
Simulation Time	180 s
MAC Protocol	802.11 a
Channel Type	Wireless
Routing Protocol	AODV
Packet Size	536 bytes

The LSTM and DQN models are implemented using TensorFlow 2.13.1. Training and testing are performed on a PC equipped with an Intel i5-9300H CPU and an NVIDIA GeForce GTX 1650 (4GB) GPU. The key hyperparameters of the DQN are listed in Table 3.

Table 3. DQN hyperparameters.

Parameters	Value
Average Step	2×10^4
Episode	50
Discount Factor	0.95
Learning Rate	1×10^{-4}
Experience Replay Buffer Size	2×10^5
Target Model Update Rate	1×10^{-2}

4.1. Convergence Evaluation

We first analyze the training convergence of the proposed TCP-RLA algorithm.

Figure 2a shows the training reward per episode. The reward generally increases during training and stabilizes at approximately 900. This stable convergence is achieved by using a linear annealing ϵ -greedy policy, which balances exploration and exploitation during DQN training.

Figure 2b,c show the throughput and RTT values across training episodes, respectively. Throughput increases as training proceeds and stabilizes at around 0.9 Mbps. Meanwhile, the average RTT decreases from an initial value of 11 milliseconds to approximately 7 milliseconds and then remains stable.

These results confirm that the training process optimizes the objective function defined in Equation (1). The model successfully improves throughput while reducing latency, thus enhancing overall communication quality.

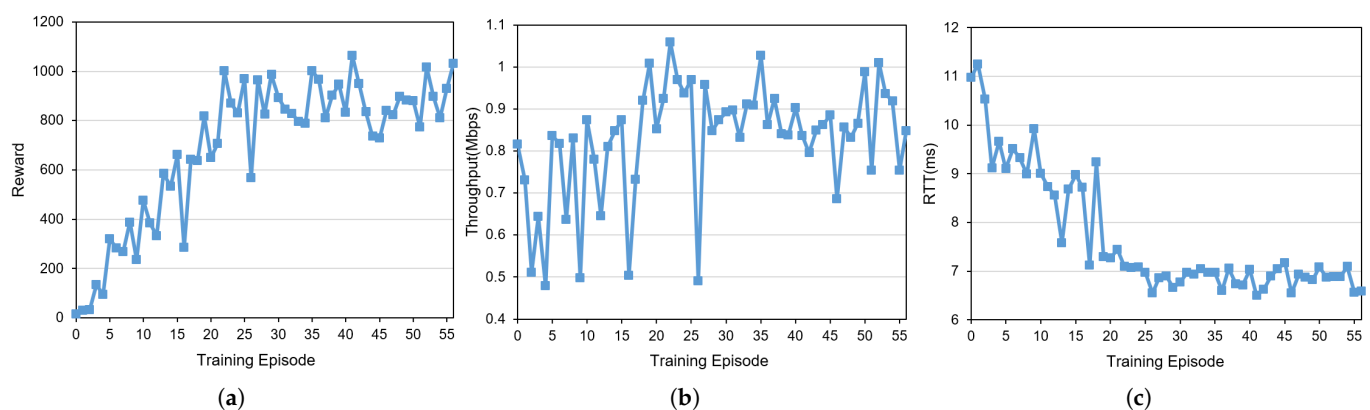


Figure 2. Convergence behavior of TCP-RLA during training. (a) Reward of TCP-RLA versus the training episode. (b) Throughput of TCP-RLA versus the training episode. (c) RTT of TCP-RLA versus the training episode.

4.2. Ablation Experiments

To verify the effectiveness of the designed prediction module, we evaluate its forecasting accuracy and compare it with the RL-only baseline.

Figure 3 shows the RTT forecasting accuracy of the LSTM-based prediction module in multi-hop ad hoc networks. By comparing predicted values with the ground truth over the entire simulation, the LSTM achieves a forecasting accuracy of 96.47%. This result demonstrates that the prediction module can effectively provide reliable network state variation information for the RL module.

To further quantify the gain introduced by the prediction module, we compare the RL-only baseline with the full scheme (RL + prediction module). The results are presented in Table 4.

The full scheme achieves higher throughput and lower RTT while maintaining a similar packet loss rate. This indicates that the prediction module effectively improves the congestion control performance of the original RL-based model.

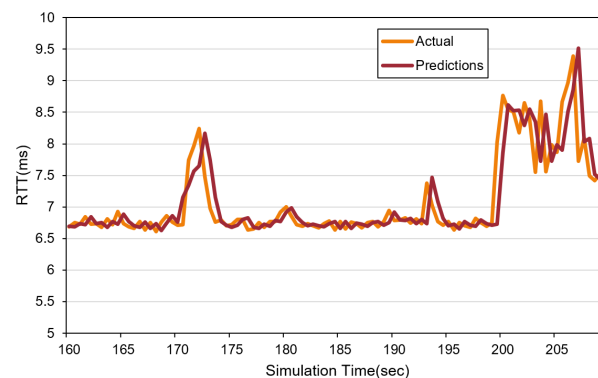


Figure 3. RTT prediction accuracy of the LSTM in multi-hop ad hoc networks.

Table 4. Ablation study results.

Metrics	Type	Value
Throughput (Mbps)	Baseline	0.832
	Baseline+Prediction	0.891
RTT (ms)	Baseline	7.00
	Baseline+Prediction	6.78
PLR (%)	Baseline	0.0749
	Baseline+Prediction	0.0727

4.3. Congestion Control Performance

The overall congestion control performance is shown in Figures 4 and 5. The results demonstrate that the trained TCP-RLA achieves strong performance under various scenarios. The model is trained under the RandomWalk2d mobility model with six nodes.

Figure 4 shows performance metrics with an increasing number of hops (RandomWalk2d, $n = 6$). Figure 4a presents throughput as the hop count increases. Throughput decreases for all algorithms as the hop count rises, but TCP-RLA maintains the highest throughput among all methods. This is because TCP-RLA's RL agent dynamically adjusts the congestion window (CWND) according to real-time network conditions, improving bandwidth utilization while alleviating congestion.

Figure 4b shows the RTT variation with an increasing number of hops. RTT increases for all algorithms, but TCP-RLA maintains the lowest RTT. This benefit comes from the adaptive congestion coefficient β , which allows fine-grained CWND adjustment suitable for multi-hop ad hoc environments.

Figure 4c illustrates the packet loss rate (PLR) under different hop counts. At two hops, all algorithms achieve a PLR of 0 due to light congestion. As the hop count increases, TCP-RLA still achieves the lowest PLR. The prediction module enables the DQN to anticipate network changes, thereby avoiding congestion and reducing packet loss.

We further compute the average performance improvement over all hop counts. Compared with TCP-GVegas and TCP-Orca, TCP-RLA improves average throughput by 36.1% and 43.3%, respectively. It reduces average RTT by 13.1% and 47.9% and lowers PLR by 33.3% and 50%, respectively.

We also evaluate the generalization ability of TCP-RLA under unseen mobility patterns and node densities, without retraining. The results are shown in Figure 5.

Figure 5a shows throughput under different unseen mobility patterns (hop = 4, $n = 6$). All algorithms perform best under ConstantPosition due to the static topology. Even though TCP-RLA is trained under RandomWalk2d, it still achieves the highest throughput under the other three mobility patterns. The learned policy remains effective under changing mobility, and the congestion coefficient β maintains appropriate CWND adaptation.

Figure 5b shows throughput under varying node densities (hop = 4, RandomWalk2d). Throughput decreases for all algorithms as more nodes join the network. Although TCP-RLA is trained with only six nodes, it still outperforms all baselines under other node densities. The prediction module effectively captures network state changes when node density changes, allowing stable and strong performance.

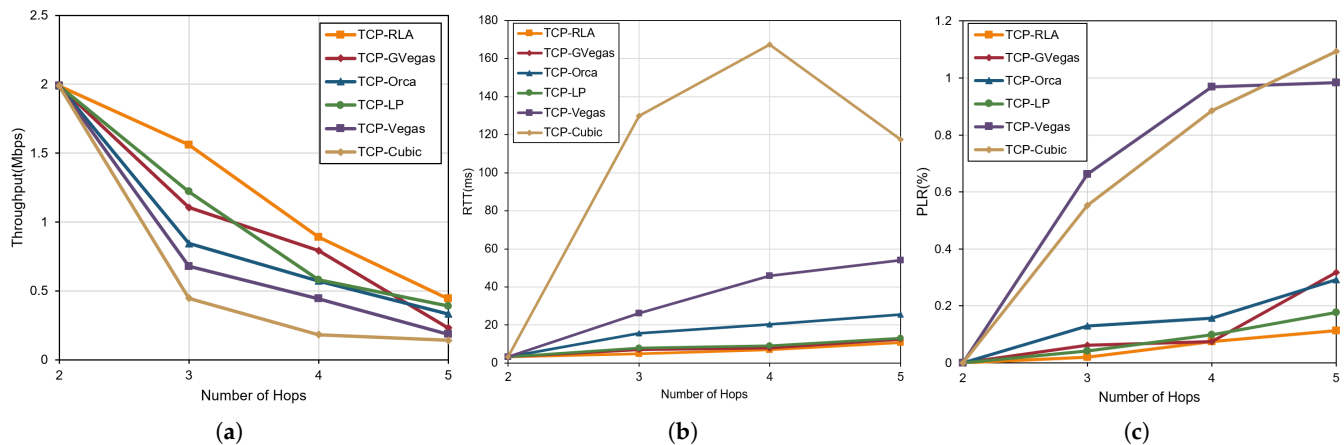


Figure 4. Performance under varying hop counts. (a) Throughput statistics for increased number of hops. (b) RTT statistics for increased number of hops. (c) PLR statistics for increased number of hops.

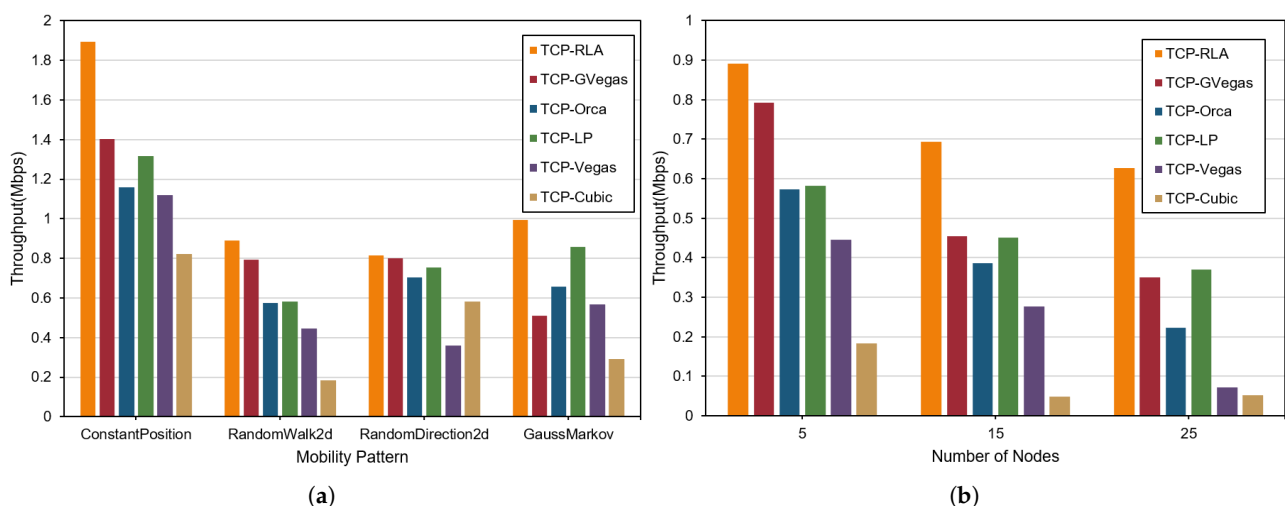


Figure 5. Generalization performance under unseen scenarios. (a) Throughput under unseen mobility patterns. (b) Throughput under unseen node densities.

4.4. Robustness Evaluation

To investigate the robustness limits of the learned policy, we introduce experiments to evaluate the learned policy under delayed, noisy, and partially observable state variables and to quantify the corresponding performance degradation relative to the ideal case. In addition, we test the model under training–testing mismatch conditions.

Figure 6 shows TCP-RLA throughput under different types of imperfect state inputs. Figure 6a shows throughput under noisy states. We add Gaussian noise to all input state

variables, which disturbs the RL agent's perception of network conditions. Throughput decreases as noise intensity increases. When the standard deviation of noise reaches 5% of the real state value, throughput drops by 11% compared with the ideal case. Thus, the noise level should be limited below 5% for stable performance.

Figure 6b shows throughput under delayed states. State delay reduces the reaction speed of the RL agent. Throughput decreases as the delay increases. When the state delay reaches 0.3 s, throughput decreases by 10% relative to the ideal case. Therefore, the state delay should be controlled within 0.3 s.

Figure 6c illustrates throughput under partially observable states. In each trial, we fix one state variable to its initial value. The results indicate that RTT most significantly influences the learned policy, followed by CWND. Among the four state variables, RTT and CWND play the most critical roles in algorithm performance.

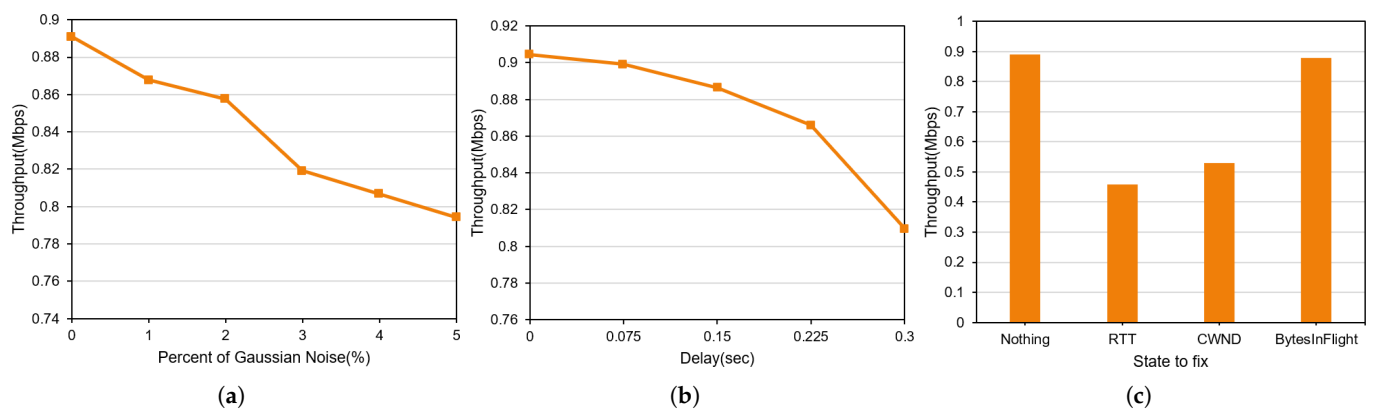


Figure 6. Throughput performance under imperfect state conditions. (a) Throughput under noisy states. (b) Throughput under delayed states. (c) Throughput under partially observable states.

We also evaluate training–testing mismatch using bursty heterogeneous traffic. The model is trained under fixed traffic with only long-lived flows but tested under mixed traffic with both long-lived and short-lived flows. Short flows arrive randomly and in bursts during the simulation. Figure 7 shows performance under this bursty scenario. As the survival duration of short flows increases, throughput decreases while RTT and PLR increase. When short flows intensify network congestion, TCP-RLA proactively reduces CWND. This adaptive mechanism prevents excessive performance degradation under heavy and bursty traffic.

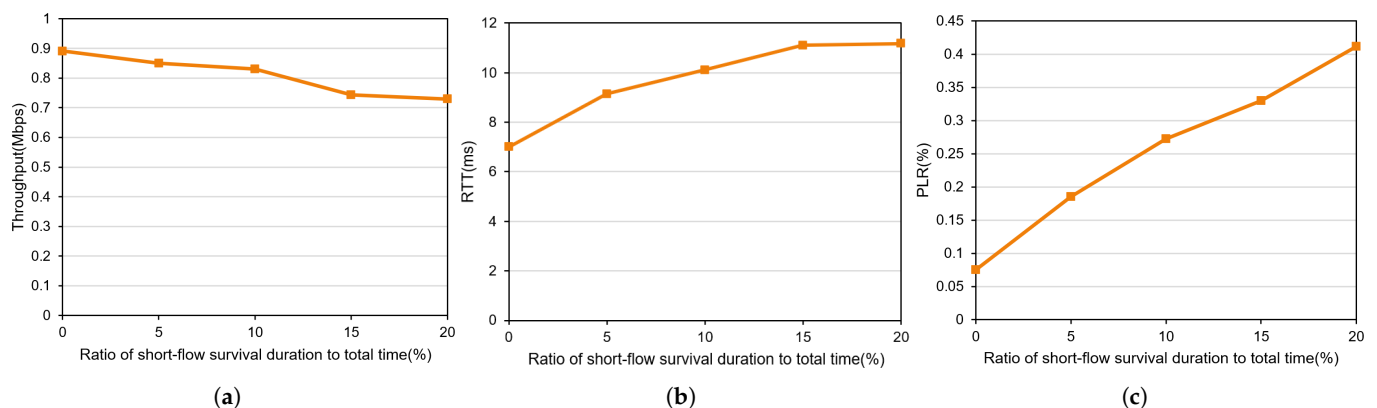


Figure 7. Performance under bursty heterogeneous traffic. (a) Throughput under increasing short-flow duration. (b) RTT under increasing short-flow duration. (c) PLR under increasing short-flow duration.

4.5. Overhead

We evaluate computational overhead by measuring CPU utilization on an Ubuntu system using the `top` command. Figure 8 shows CPU usage from 1 to 30 s for TCP-RLA, TCP-GVegas, TCP-Orca, and TCP-LP. The rule-based algorithm TCP-LP shows the lowest complexity, with an average CPU utilization of 5.17%. TCP-RLA and TCP-Orca maintain relatively low overhead, with average utilizations of 17.19% and 15.77%, respectively. TCP-GVegas imposes the highest computational load, reaching 63.05%. Although TCP-RLA has higher overhead than traditional rule-based methods, it maintains low computational cost among RL-based schemes. This efficiency comes from the use of a lightweight DQN structure and a discrete action space combined with a rule-based mechanism, which reduces complexity and speeds up convergence.

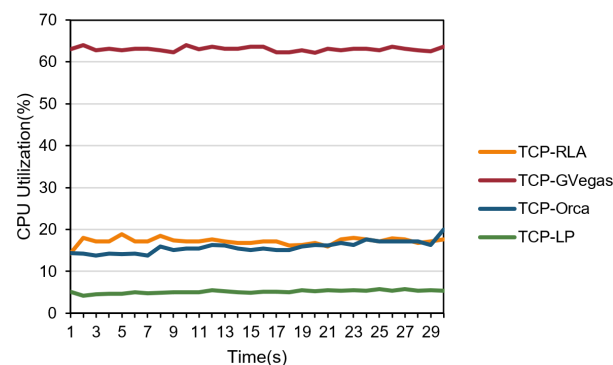


Figure 8. CPU utilization comparison between TCP-RLA and other schemes.

5. Discussion

In terms of throughput, RTT, and PLR, TCP-RLA achieves the best overall performance. This improvement comes from three key design choices. First, the reinforcement learning agent in TCP-RLA adjusts the congestion window (CWND) according to real-time network conditions. It maximizes bandwidth utilization while avoiding excessive congestion. Second, the prediction module forecasts critical network state parameters. This enables the DQN to make more forward-looking and robust decisions. Third, the customized action space is well-suited to the dynamic conditions of multi-hop ad hoc networks.

By comparison, other algorithms have clear limitations. TCP-GVegas enhances TCP Vegas using a gray prediction model to estimate future throughput. It also relies on a cross-layer framework to compute expected throughput. Although performance is improved, the method depends heavily on cross-layer information and manual threshold tuning. TCP-Orca uses a two-layer control structure: a rule-based module for fine-grained control and a DRL agent for coarse-grained updates. However, this structure cannot fully adapt to frequent topology changes in multi-hop ad hoc networks. For these reasons, TCP-RLA outperforms existing approaches.

In addition to performance gains, TCP-RLA maintains low computational complexity. This makes it suitable for deployment in resource-constrained ad hoc networks.

Although TCP-RLA performs well in NS-3 simulations, real-world environments are more dynamic and unpredictable. We therefore conducted a thorough robustness evaluation. Experimental results show that performance degrades under delayed, noisy, or partially observable states. To address these challenges, we propose several potential mitigation strategies:

1. Apply exponential moving average (EMA) filtering to smooth noisy state inputs;
2. Use recurrent structures such as LSTM to infer hidden states from delayed observations and train with randomized delays to improve robustness;

3. Introduce attention mechanisms to assign higher weights to critical features such as RTT and CWND.

Finally, the current version of TCP-RLA optimizes performance metrics for each individual TCP connection. The fairness among multiple competing TCP flows in ad hoc scenarios remains an important direction for future investigation.

6. Conclusions

In this paper, we propose a lightweight adaptive reinforcement learning algorithm, termed TCP-RLA. The core novelty of this work resides in integrating an LSTM-based future network state prediction module into the reinforcement learning framework. This integration enables the algorithm to make more foresighted and proactive decisions. In scenarios with multiple TCP connections, TCP-RLA can proactively reduce the congestion window (CWND) when network resources are exhausted. Conversely, it increases CWND during periods of network resource idleness. Furthermore, we devised a dedicated action space tailored to multi-hop ad hoc networks. This action space can adaptively adjust to dynamic network congestion levels. It also accelerates algorithm convergence and reduces computational complexity. In addition, we put forward an optimization formulation (Equation (1)). This formulation incorporates key performance indicators (KPIs) relevant to multi-hop ad hoc networks. The reward function of TCP-RLA is constructed based on this optimization formulation.

As demonstrated by the simulation results in Section 4, TCP-RLA outperformed the other two RL-based algorithms (TCP-GVegas and TCP-Orca). Compared with TCP-GVegas and TCP-Orca, TCP-RLA achieved an average throughput improvement of 36.1% and 43.3%, respectively; a mean reduction in round-trip time (RTT) of 13.1% and 47.9%, respectively; and an average packet loss rate (PLR) reduction of 33.3% and 50%, respectively.

Author Contributions: Conceptualization, H.L.; methodology, H.L.; software, Z.X.; validation, H.L. and Z.X.; writing—original draft preparation, Z.X.; writing—review and editing, H.L.; visualization, Z.X.; supervision, H.L.; project administration, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Agrawal, R.; Faujdar, N.; Romero, C.A.T.; Sharma, O.; Abdulsahib, G.M.; Khalaf, O.I.; Mansoor, R.F.; Ghoneim, O.A. Classification and comparison of ad hoc networks: A review. *Egypt. Inform. J* **2023**, *24*, 1–25.
2. Cao, C.; Tian, Z. Survey on TCP Congestion Control in Ad Hoc Wireless Networks. In Proceedings of 2024 36th Chinese Control and Decision Conference (CCDC), Xi'an, China, 25–27 May 2024; pp. 3230–3235.
3. Mast, N.; Khan, S.; Uddin, M.I.; Ghadi, Y.Y.; Alkahtani, H.K.; Mostafa S.M. A Cross-Layer Solution for Contention Control to Enhance TCP Performance in Wireless Ad-Hoc Networks. *IEEE Access* **2023**, *11*, 24875–24893.
4. Zhang, T.; Mao, S. Machine Learning for End-to-End Congestion Control. *IEEE Commun. Mag* **2020**, *58*, 52–57.
5. Maliha, M.; Habibi, G.; Atiquzzaman, M. A Survey on Congestion Control and Scheduling for Multipath TCP: Machine Learning vs Classical Approaches. In Proceedings of 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS), Warsaw, Poland, 17–20 September 2023; pp. 49–61.
6. Jiang, H.; Li, Q.; Jiang, Y.; Shen, G.; Sinnott, R.; Tian, C.; Xu, M. When machine learning meets congestion control: A survey and comparison. *Comput. Netw.* **2021**, *192*, 108033.

7. Giacomoni, L.; Parisi, G. Reinforcement Learning-based Congestion Control: A Systematic Evaluation of Fairness, Efficiency and Responsiveness. In Proceedings of IEEE INFOCOM 2024—IEEE Conference on Computer Communications, Vancouver, BC, Canada, 20–23 May 2024; pp. 1451–1460.
8. De La Fuente, N.; Guerra, D.A.V. A comparative study of deep reinforcement learning models: DQN vs PPO vs A2C. *arXiv* **2024**. Available online: <https://arxiv.org/abs/2407.14151> (accessed on 4 November 2024).
9. Yan, F.Y.; Ma, J.; Hill, G.D.; Raghavan, D.; Wahby, R.S.; Levis, P.; Winstein, K. Pantheon: The training ground for Internet congestion control research. In Proceedings of 2018 USENIX Annual Technical Conference (USENIX ATC 18), 11–13 July 2018; pp. 731–743.
10. Molia, H.K.; Kothari, A.D. TCP-RLLD: TCP with reinforcement learning based loss differentiation for mobile adhoc networks. *Wirel. Netw.* **2023**, *29*, 1937–1948.
11. Pillai, B.; Chhabra, G. TCP-CNNLSTM: Congestion Control Scheme for MANET using AI Technologies. In Proceedings of 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), Trichy, India, 23–25 August 2023; pp. 63–69.
12. Ali, M.H.; Ozturk, S. Efficient congestion control in communications using novel weighted ensemble deep reinforcement learning. *Comput. Electr. Eng.* **2023**, *110*, 108811.
13. Jay, N.; Rotman, N.H.; Godfrey, P.B.; Schapira, M.; Tamar, A. A Deep Reinforcement Learning Perspective on Internet Congestion Control. In Proceedings of International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019.
14. Mahajan, S.; Sridevionmalar, P.; Harikrishnan, R.; Kotecha, K. Reinforcement Learning Based Congestion Control Technique for Wireless Mesh Networks. *Int. J. Netw. Distrib. C* **2025**, *13*, 20.
15. Jiang, H.; Luo, Y.; Zhang, Q.; Yin, M.; Wu, C. TCP-Gvegas with prediction and adaptation in multi-hop ad hoc networks. *Wirel. Netw.* **2017**, *23*, 1535–1548.
16. Abbasloo, S.; Yen, C.Y.; Chao, H.J. Classic meets modern: A pragmatic learning-based congestion control for the Internet. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 632–647.
17. Alipio, M.; Bures, M. A cache-aware congestion control mechanism using deep reinforcement learning for wireless sensor networks. *Ad Hoc Netw.* **2025**, *166*, 103678.
18. Li, M.; Huang, X.; Jin, C.; Pei, Y. A TCP Congestion Control Algorithm Based on Deep Reinforcement Learning Combined with Probe Bandwidth Mechanism. In *Proceedings of the 5th International Conference on Computer Science and Application Engineering (CSAE '21)*; Association for Computing Machinery: New York, NY, USA, 2021; Article 44, pp. 1–7.
19. Goyal, P.; Rishiwal, V.; Negi, A. A comprehensive survey on QoS for video transmission in heterogeneous mobile ad hoc network. *Trans. Emerg. Telecommun. Technol.* **2023**, *34*, e4775.
20. Anwekar, D.; Phulre, S. Analysis of Congestion Control Techniques to Improve QoS and Frequent Communication in FANET. In Proceedings of 2023 World Conference on Communication and Computing (WCONF), RAIPUR, Raipur, India, 14–16 July 2023; pp. 1–7.
21. Kuzmanovic, A.; Knightly, E.W. TCP-LP: Low-priority service via end-point congestion control. *IEEE ACM Trans. Netw.* **2006**, *14*, 739–752.
22. Hemmati, A.; Zarei, M.; Rahmani, A.M. A systematic review of congestion control in internet of vehicles and vehicular ad hoc networks: Techniques, challenges, and open issues. *Int. J. Commun. Syst.* **2024**, *37*, e5625.
23. Grover, A.; Kumar, R.M.; Angurala, M.; Singh, M.; Sheetal, A.; Maheswar, R. Rate aware congestion control mechanism for wireless sensor networks. *Alex. Eng. J.* **2022**, *61*, 4765–4777.
24. Gawłowicz, P.; Zubow, A. ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '19)*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 113–120.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.