Nuwa-RL: A Reinforcement Learning based Receiver-side Congestion Control Algorithm to Meet Applications Demands over Dynamic Wireless Networks

Guanghui Gong, Xianliang Jiang, Guang Jin, Yi Xie, Haiming Chen
Ningbo University
Ningbo, Zhejiang, China
{2011082292, jiangxianliang}@nbu.edu.cn

***Abstract*—The advent of the wireless network has significantly increased the amount of data transmitted on networks. Subject to users' location and network conditions, data transmitted on networks exhibit different transmission characteristics and is accompanied by the problem of data transmission and network switching. Congestion Control is the key to solving these problems. However, the traditional Congestion Control is deployments relied on rule-based heuristics and tested on a predetermined set of benchmarks. Consequently, these Congestion Control algorithms cannot cope with the challenges of diversified services. To address these problems, we propose Nuwa-RL, which integrates the reinforcement learning algorithm PPO-LSTM in combination with the congestion control algorithm at the receiver side to fulfill control over network data transmission. Based on data obtainable at the receiver side, Nuwa-RL dynamically adjusts changes in the window to fulfill control over data transmission by using reinforcement learning. Experiments show that PPO-LSTM is more suitable for use in wireless networks. Compared to traditional congestion control methods, Nuwa-RL achieves high throughput with low latency.**

***Index Terms*—receiver-side, Congestion Control, reinforcement learning**

I. Introduction

The main factor of network congestion is the speed at the data senders to transmit data exceeds the processing of the receiver, result the data buildup in the network causing congestion. Network congestion is accompanied by packet loss and increased transmission delay. These problems are intolerable in applications that require high throughput and low latency. Therefore, providing a stable network service and reliable data transmission is crucial for the current network.

Congestion control is the main component of current networks and the key method used to solve network congestion problems. However, the traditional Congestion Control is deployments rely on rule-based heuristics and tested on a predetermined set of benchmarks. As a result, traditional congestion control algorithms are not performing as well as they could when the network becomes more complex. And more granular control is needed in existing networks. Therefore, designing a congestion control algorithm that can handle network challenges is critical for data transmission performance.

With the development of Reinforcement Learning(RL), researchers have found that using RL to solve congestion problems is a very promising solution. Such as Q-TCP [16], uses Q-learning algorithm to design a congestion control algorithm. It helps the sender to learn optimal congestion control in an online manner without any prior knowledge of the network model. Kong et al [2] proposed two learning-based TCP congestion control schemes for wired networks. One is a supervised learning-based packet loss predictor and the other is a congestion control scheme based on RL, namely SARSA algorithm.

In this work, the aim is to design a congestion control algorithm that adapts to changes in the network and responds to data transmission in a targeted manner. We implemented an algorithm called Nuwa. In Nuwa, we implement a decoupled congestion avoidance phase on the sender side and implement it on the receiver side. Compared to other traditional congestion control algorithms, Nuwa achieves 20% throughput improvement. However, Nuwa is a heuristic-based algorithm that can only make decisions for a specific network state. When the network state is constantly changing, Nuwa cannot adapt to all network states. When the network switchover, Nuwa is prone to throughput degradation and delay increase. Therefore, we propose Nuwa-RL to improve the heuristic congestion control algorithm using RL so that Nuwa-RL can be adapted to different network environments.

In this paper,we make the following contributions:

- Based on the Nuwa algorithm we implement an RL version of the algorithm (Nuwa-RL), and propose an RL-based congestion control scheme for the receiver side.

Manuscript received xx xx, 2022. This work was supported in part by the National Natural Science Foundation of China (61601252), the Natural Science Foundation of Zhejiang Province (LY20F020008, LY21F020006), the Ningbo Natural Science Foundation (202003N4085), the Ningbo Public Welfare Project (202002n3109), the Special Research Funding from the Marine Biotechnology and Marine Engineering Discipline Group in Ningbo University (No. 422004582), the Ningbo Key Science and Technology Plan (2025) Project (2019B10125, 2019B10028, 20201ZDYF020077).

- We test multiple RL methods and the combined RL-based scheme in a dynamic network environment to verify the effectiveness of Nuwa-RL in adapting to dynamic network environments.

The structure of this paper is shown as follows. Section 2 introduces the background. Section 3 illustrates the problem. Section 4 presents the feasibility of using RL. Section 5 details the implementation of the Nuwa-RL framework, and the experimental justification in Section 6. Section 7 presents related work and concludes the full paper in Section 8.

II. Background

A. Traditional TCP Congestion Control

The purpose of congestion control is to achieve rational transmission of packets in the TCP network. Under the premise of fully utilizing the network bandwidth, the sending and receiving ends negotiate the data transmission range through congestion control to achieve reasonable utilization of network resources. The basic idea is to maintain two windows, the congestion window (Cwnd) and the receive window (Rwnd), at the sending and receiving ends, respectively. Data transmission is regulated by selecting the smallest of the two windows as the send window (Swnd) for data transmission. For each link, Cwnd is able to dynamically determine the total number of packets to be transmitted. Therefore, congestion control is able to achieve the control of data transmission.

Most of the current congestion control schemes are implemented based on heuristic algorithms, such as CUBIC, Vegas, etc. The implemented congestion control is divided into four main phases (the Rwnd uses the default adjustment policy), namely Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery.

Slow Start. Cwnd is set to an MSS (Maximum Message Segment) at the beginning of a TCP connection. The congestion window is doubled whenever the TCP sender sends all the data in the sending window and receives all the acknowledgments successfully, i.e., the window changes exponentially during this phase to occupy the available bandwidth quickly.

Congestion Avoidance. When the window is larger than ssthresh (the dividing point between the slow start phase and the congestion avoidance phase of congestion control). Congestion control enters the congestion avoidance phase. Cwnd is set to 1 when the network is congested, and ssthresh is set to half the size of the congestion window when congestion occurs, then congestion control reverts to the slow start phase.

Fast Retransmit. This phase is proposed to enable the sender to retransmit packets as soon as possible in case of packet loss, instead of waiting for the timeout timer to be updated, and TCP requires the receiver to send a reception acknowledgment immediately after receiving data. Instead of waiting for the timeout timer to expire, Fast Recovery.

Fast Recovery. This phase typically occurs in conjunction with Fast Retransmission. The current packet is considered lost when the sender receives three consecutive duplicate acknowledgments, and this phase sets both ssthresh and Cwnd to half of the congestion window in the current state, and then enters the congestion avoidance phase.

The traditional Congestion Control deployments rely on rule-based heuristics to provide feedback on specific network signals to regulate data transmission. At the same time this also brings some problems: (1) it may not be able to adapt to the new network environment when the network changes rapidly; (2) the traditional congestion control scheme has a single form of control, and once the function is determined, the corresponding policy is not capable of self-optimization. The above two drawbacks lead to the inability of the traditional congestion control algorithm to achieve the promised results in the current networks.

B. RL-based Congestion Control

To address the shortcomings of traditional congestion control algorithms, research has focused on RL-based congestion control algorithms. It can adjust the control policy immediately according to the change in network status. Existing RL-based congestion control systems can be divided into two categories: the value-based and the policy-based RL congestion control systems.

a) Value-based RL Congestion Control System: Value-based RL algorithms make policy decisions by using the introducing value functions to estimate the expected returns at future moments. The core of value-based algorithms lies in how to optimize the value function by continuously exploring the environment, so as to establish a metric for the decision payoff of the entire state-action space, and use it as a basis for policy decisions. The value function contains two forms: (1) State-value function: given the current policy$\pi(\mathrm{a} \mid \mathrm{s})$,Use $\mathrm{V}^{\pi}(s) = \sum_{s_t, a_t \sim \pi} [\sum \gamma^t r(S_t, a_t) \mid s_0 = s]$ to indicate the mapping of state space to actual markers. (2) State-action value function: given the current policy$\pi(\mathrm{a} \mid \mathrm{s})$, $\mathrm{Q}^{\pi}(s, a) = \sum_{s_t, a_t \sim \pi} [\sum \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a]$. It is easy to see that the two functions satisfy $\mathrm{V}^{\pi}(s) = \sum_a Q^{\pi}(s, a)\pi(a \mid s)$. When the state-action function under the optimal policy is solved, it can be used for policy-making. In the case that the environment model is determined, it can be solved using dynamic programming-based value iteration or policy iteration methods. Current value-based optimization methods include both Monte Carlo methods and Temporal Difference learning schemes.

The Monte Carlo method approximates the value of the function mainly by sampling the transfer model. In each round of stochastic exploration, a random initial state is started. The algorithm continuously makes greedy decisions using the current state-action function until the end of the decision sequence.

The Temporal Difference method is mainly used when the search range of states or actions is large. In this case, the Monte Carlo method is significantly constrained in terms of sample complexity and training efficiency due to the limitation that the complete trajectory must be sampled after each round. To solve this problem, Richard S. Sutton et al. proposed the method of Temporal Difference (TD) learning to increase

the frequency of value function updates. The basic idea of Temporal Difference learning is to use the existing value function estimates as part of the value function supervised signal, to avoid the problem of having to sample to the end state of the trajectory in order to perform calculations on the returns. Representative algorithms include Q-learning, DQN, and other algorithms.

b) Policy-based RL Congestion Control System: RL methods such as Monte Carlo methods and temporal differencing use the parameters to approximate the target value, and the actions taken by these methods are determined at each state. However, the limitations of the individual in observing the environment may lead to similar environments where different actions should be taken. In this case, the preferred strategy is stochastic, where the actions taken need to be different and actions that approximate the deterministic value are flawed. Therefore, a Policy-based RL algorithm is proposed.

The objective of the policy-based approach is to parameterize the policy. It will build the policy model $\pi_\theta(s, a)$. When the input states, the algorithm is able to calculate the probability of each action being executed to obtain a high return and selects the action according to the probability. θ is the key to obtaining the optimal strategy $\pi_\theta(s, a)$. The policy function is shown in equation 1.

$$\pi_\theta(s, a) = P(a \mid s, \theta) \approx \pi(a \mid s) \tag{1}$$

The policy function $\pi_\theta(s, a)$ is a probability density function that determines the probability of all behaviors by given states and certain parameter settings. And the parameter θ determines the specific form of the strategy, so it is necessary to solve the optimal parameter θ and thus decide the optimal strategy through the objective function $J(\theta)$. The current main objective function $J(\theta)$ has three forms, as shown in equation (2)(3)(4).

$$J_1(\theta) = V_{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}(G_1) \tag{2}$$

$$J_{avV}(\theta) = \sum_s d_{\pi_\theta}(s) V_{\pi_\theta}(s) \tag{3}$$

$$J_{avR}(\theta) == \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(s, a) R_s^a \tag{4}$$

The Eq. (1) can be used to solve for $J(\theta)$ when there is a well-defined initial state for the optimization objective. Eq. (2) is used when the optimization objective does not have an explicit initial state, then the average value is used to optimize the objective. Eq. 3 then calculates $J(\theta)$ by defining the average reward for each step.

By [4] we can derive that the gradient of the above three equations with respect to the optimization objective can be expressed as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta}\left[\nabla_\theta \log \pi_\theta(s, a) Q_\pi(s, a)\right] \tag{5}$$

With the optimization objective of the function, by analyzing a specific scenario, we are able to define the corresponding evaluation function.

III. Nuwa design and problems

A. Nuwa Algorithm

We implemented a heuristic-based congestion control algorithm, named Nuwa. The algorithm is implemented on the receiver side by decoupling the congestion avoidance phase from the congestion control algorithm on the sender side. When entering the congestion avoidance phase, we modify the code on the sender side so that the sender set the Cwnd as the receive window from the receiver's reply ack header. The control logic of the receive window at the receiver side is also modified to adjust the receive window using a heuristic algorithm. Nuwa implements optimization of the traditional congestion control based on the Linux kernel control logic to achieve improved data throughput in a wireless network environment. The receiving window adjustment logic in Nuwa is shown below:

- First, Nuwa calculates the current network one-way delay based on equation (6) to determine the degree of network link usage.

$$\mathrm{D}_c = (S_t - S_m) - (R_t - R_m) \tag{6}$$

- Then set different experimental targets according to the characteristics of different links and user usage scenarios T_d
- Considering the mutability of the network, we use $\theta = \tan h(x)$ as the adjustment function of the Rwnd. Where $\mathrm{x} = \frac{T_\mathrm{d} - D_\mathrm{c}}{\rho}$. The positive and negative x's are used to control the direction and magnitude of adjustment of the window. ρ is the sensitivity factor for queuing delay fluctuations, which is used to map the experimental network changes into the tanh function to represent the network changes.
- Finally, use equation (7) to adjust the receive window size. The formula enables precise control of the window in different network situations. When the network changes a lot, the window is at a high level, and it will reduce the size of the window adjustment range, and vice versa.

$$w_{\text{new}} \leftarrow w_{\text{old}} + (\theta \cdot k)/w_{\text{old}} \tag{7}$$

B. Limit of Nuwa

Nuwa is able to effectively prevent throughput degradation due to bursts of network degradation in networks. However, Nuwa is a heuristic-based dominance control algorithm. When the network network switching will result that the parameter is not suitable for the changed network, resulting in failure to achieve the promised results.

Since the popularization of 5G, there is a situation where 4G and 5G networks are shared in the current network. According to the International Telecommunication Union (ITU), the 4G

transmission rate reaches 1Gbps and can reach 100Mbps in a high-speed mobile state. 5G technology has further improved the data transmission capability compared to 4G. By using a higher frequency band based on 4G, 5G makes the transmission more efficient. The peak transmission speed of 5G can reach 20Gbps, which is more than 20 times faster than the transmission speed of 4G networks. But the increase in waveband also means less base station coverage. This makes the problem more obvious when switching between networks in life. We experiment in real mobile networks to test the performance of Nuwa in different network environments and compare it with the traditional congestion control algorithm CUBIC.

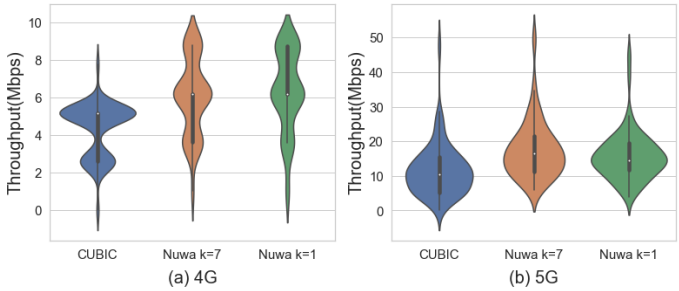


Fig. 1. Throughput in Nuwa's different networks.

The experimental results are shown in Figure 1. The k is a key value in Nuwa that can adjust the aggressiveness of the algorithm window change. Therefore, in our experiments, we used different values of k in 4G and 5G networks. According to the results, Nuwa has a higher improvement in network throughput than the traditional congestion control algorithm. In 4G networks, it is more obvious that Nuwa has improved the data transfer speed. In contrast, the transmission speed of CUBIC is stable in two ranges of 3Mbps and 5Mbps. Nuwa's transmission speed to the traditional network is around 5Mbps in 5G network.

The effect of the k value on Nuwa is mentioned in the distribution of transmission speed. In 4G networks, a small value of k is more conducive to windowing because of the low bandwidth. It can avoid excessive window adjustment. From 2(a), it shows that the data distribution of k=1 is more consistent with the 4G network when comparing the different upper and lower quartile positions. And with 5G networks, a more aggressive tuning scheme is required due to the higher jitter of 5G networks. When k=7, Nuwa is more compatible with the 5G network environment.

C. Why PPO-LSTM

Based on previous experiments, selecting a suitable k value for a TCP stream in Nuwa is not an easy task. Using the traditional heuristic for the k value is the current Nuwa main solution, and it is the main way for traditional congestion control to adjust the key value. However, with the increase in network complexity, traditional heuristic schemes cannot achieve matching for multiple network environments. The complexity of the network is reflected in its high correlation between multiple complex factors and its emphasis on real-time. Such as bandwidth size, RTT, current sending window size, etc. Therefore, it is difficult to select the appropriate value in a large search range. However, matching in large search ranges is something that RL excels at. RL, inspired by the psychology of human behavior [5], is a popular technique in the machine learning community. RL constantly makes decisions based on environmental feedback. Once the reward function achieves a dynamic balance between exploring suboptimal decisions and exploiting the current optimal decision, it is able to find the best decision based on trial and error. Therefore, RL is a fit for tuning the parameters in the algorithm according to the real-time changes in the network.

In this work, we select PPO-LSTM as the main algorithm to improve Nuwa. PPO is a policy gradient algorithm proposed by Schulman et al. in 2017. PPO is an improvement on Trust Region Policy Optimization (TRPO), which uses a simpler tailoring agent target and omits the expensive second-order optimization in TRPO. But PPO involves the learning and computation process of updating with minimally cropped strategies at each iteration. The large differences between the old and new strategies easy to make wrong decisions. [7] proposes that the inclusion of LSTM in the network of PPO can effectively improve the prediction effect. And we believe that the addition of LSTM can even solve the learning problem of PPO in the network effectively, our subsequent experiments demonstrate this. Therefore, we introduce PPO-LSTM in Nuwa-RL to solve the problems encountered in Nuwa.

IV. Nuwa-RL Preview

A. Nuwa-RL Design

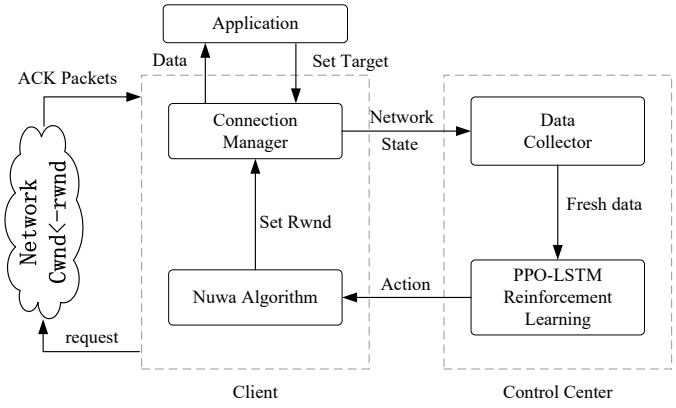


Fig. 2. Nuwa-RL Design.

We add the RL part to Nuwa to implement our design, and the overall architecture diagram is shown in Figure 2. The design is divided into three parts: (1) server-side configuration, (2) receiver-side configuration, (3) and the RL part.

The **server-side** is mainly used to provide the data support services required by the client. We mainly modified the con-

gestion avoidance part of the kernel on the server side, and we handed over the functionality of that part to the receiver side. The server side set the window value in the client-side packet header as the current send window size. As **receiver-side**, we deploy the Nuwa-RL algorithm in the kernel. With 10 lines of code, we export the tcp_clamp_window method in kernel tcp_input.c and implement the same algorithm deployment as server-side congestion control with module plugging. We have implemented **reinforcement learning** using the Stable Baselines3 framework in the user state. Since the kernel state and the user state cannot communicate directly in Linux, we redesigned the data collection part and the data transfer part. First, we implemented the real-time collection of kernel data using the eBPF technique. The eBPF collects the kernel data through kprobe technology and efficiently delivers the data to the user state. Considering the relevance to the current form of algorithm deployment at the receiver side, we use Neilink for passing data to the kernel. This technique uses socket technology to pass data between the user state and the kernel state. We present the main algorithm of Nuwa-RL in Algorithm 1.

Algorithm 1 The Nuwa-RL Algorithm

1: $initial:$
2: $T_d \leftarrow 0, S_m \leftarrow 500\ (ms), R_m \leftarrow 0;$
3: $Delay_{min} \leftarrow a\ large\ value;$
4: $\rho \leftarrow a\ suitable\ value$
5: **for** each ACK **do**
6: $S_t \leftarrow Timestamp\ of\ packet\ delivery;$
7: $R_t \leftarrow Timestamp\ of\ packet\ receive;$
8: $D_c = (S_t - S_m) - (R_t - R_m)$
9: **if** $Delay_{min} = 0 || D_c < Delay_{min}$ **then**
10: $S_m \leftarrow S_t$
11: $R_m \leftarrow R_t$
12: $Delay_{min} \leftarrow D_c$
13: **end if**
14: x $\leftarrow \frac{T_d - D_c}{\rho}$
15: $\theta \leftarrow tanh(x)$
16: $k \leftarrow get\ k\ from\ Reinforcement\ Learning$
17: $w_{new} \leftarrow w_{old} + (\theta \cdot k)/w_{old}$
18: **end for**

Considering that different systems have different clock frequencies, we use the one-way delay in the link to calculate the current network state. We collect the current data sending time and arrival time by collecting each ack packet information and use equation (1) for the one-way delay (lines 6-8). The accuracy of the D_c data is ensured by recording the minimum transmission delay of the current network packet (lines 9-12). We compute the previously collected data and also obtain the k-value returned by RL part and bring it into Eq. (7) to compute the new Rwnd.

B. RL Algorithm Design

The States. The state information in PPO-LSTM is used for inference of actions and learning of models. An accurate, real-time state is a prerequisite for RL algorithms to make correct decisions. Using the knowledge from past can effectively improve the performance of the algorithm. Therefore, a trade-off between model convergence and state space needs to be considered. Based on the appeal considerations, we selected the following characteristics as the data for state selection:

- rcv_data: the data received by the receiver during a period of time.
- new_win: new window size
- racv_space: The size of the real-time cache on the receiver side.
- rtt_us: rtt of the last packet sent

With the eBPF technique, we obtain real-time data from the kernel as input to the model State.

Actions. In the Nuwa algorithm, the k value is mainly adjusted for the radicality of the window change. Therefore, according to the range of window changes, we classify the actions of RL output into five types, matching different levels of changes. Inspired by the BBR [15] algorithm, using approximately equal to 2.89 as one of the behaviors allows the agent to quickly adapt to changes in the network environment. To achieve the optimal sending rate, the sending rate is fine-tuned using 1.25 and 1.05, respectively. After the algorithm reached stability action=1 was able to keep the behavior stable at the optimal operating point. When the network state starts to deteriorate, we make $action = 0.25$ to achieve the reverse adjustment of the window to achieve the adaptation to the network state change. The Nuwa-RL action space is shown below.

$$Action = \{0.25, 1, 1.05, 1.25, 2.85\} \tag{8}$$

Reward Function. In resource allocation, the alpha-fair function [6] is widely used to evaluate the quality of bandwidth allocation when multiple streams share the same link. The alpha-fair function is defined as follows.

$$\mathrm{U}_\alpha(x) = \begin{cases} \log(x), \alpha = 1 \\ \frac{x^{1-\alpha}}{1-\alpha}, \alpha > 0, \alpha \neq 1 \end{cases} \tag{9}$$

Nuwa's implementation at the receiver-side aims to achieve a tradeoff between throughput and latency. It is important to maintain low latency and low packet loss in the network while achieving high bandwidth utilization and good fairness. Combining the alpha-fairness function we set the utility function as shown in Equation 10.

$$\mathrm{U}\left(x(t), \tau_r(t), \tau_\mathrm{l}(t)\right) = \gamma \cdot U_\alpha(x(t)) - \theta \cdot U_\alpha\left(\tau_r(t)\right) - \varphi \cdot \tau_\mathrm{l}(t) \tag{10}$$

Where γ, θ, φ are constants to indicate the importance of throughput, delay and packet loss in the Reward Function in the formula. In the experiment, we set $\gamma = 0.4, \theta = 0.4, \varphi = 0.2, \alpha = 0.6$. Adjusting the appropriate values allows the Reward Function to be adjusted in a stable range.

Training. The Agent is implemented in python and trained in a custom Gym environment for OpenAI. We train and test in

a real network transmission environment and testbed (cellsim). In cellsim, we use the traces we collect under the real network to ensure the authenticity of the experimental results. We set 800 steps as a training period to avoid overfitting of the training.

V. Evaluation

A. Experimental environment

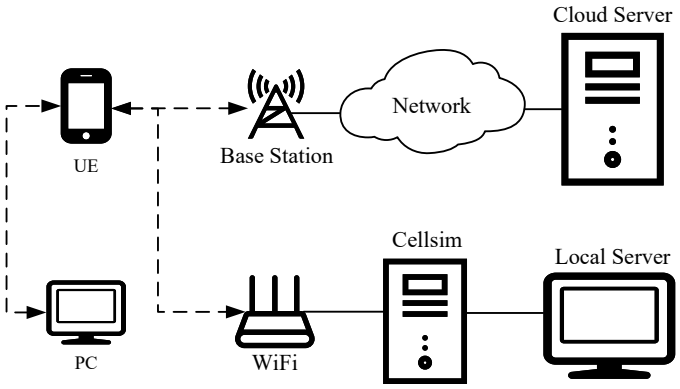


Fig. 3. Experiment Topology.

The experimental environment in this paper includes the real experimental environment and the simulated experimental environment. The topology of the experimental environment is shown in Figure 3. A Think Pad laptop X1 Carbon was used for the receiver PC to deploy the receiver algorithm and connect through the collection and wireless network. For the local network simulation experimental environment we used two mainframes (DELL i5 4G) for deploying Cellsim network simulation and as local servers, respectively. For the cloud server, we used the service provided by Huawei Cloud (4vCPU, 8GiB) for deploying cloud services. We deploy the same data on the cloud server and the local server to ensure the consistency of the experimental results.

B. RL Learning Efficiency Comparison

TABLE I
THE RL ALGORITHMS IN SB3 SUPPORT INSERT AND RESULT STRUCT.

Name	*Box*	*Discrete*	*MultiDiscrete*	*MultiBinary*
A2C	✔	✔	✔	✔
DQN	✔	✖	✖	✖
DDPG	✖	✔	✖	✖
HER	✔	✔	✖	✖
PPO	✔	✔	✔	✔
SAC	✔	✖	✖	✖
TD3	✔	✖	✖	✖

We implemented the Stable Baselines3 framework for the RL part. Different RL algorithm has restrictions on the input and output formats of the data. Therefore, we conducted a statistical analysis of the supported input and output formats of the RL algorithms implemented in the framework before our experiments. The results are shown in Table 1.

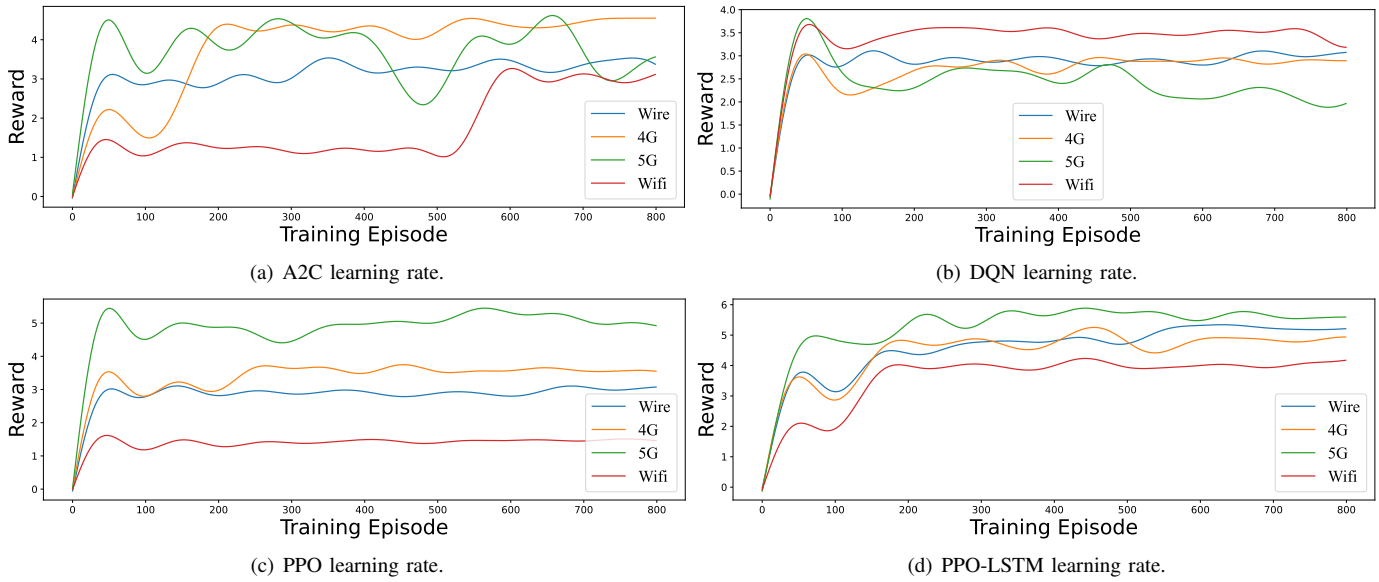
Different RL algorithm has different performances when used in wireless networks. According to table 1 and the data format in Nuwa-RL, we selected A2C, DQN, PPO, and PPO-LSTM for comparison experiments to compare the learning ability of different RL algorithms in the wireless network environment. We conducted the experiments in wired networks, WiFi, 5G, and 4G. The experimental results are shown in Figure 4.

In the experiment, we selected the learning efficiency and Reward value as the comparison condition. From the results, we found that due to the setting of the reward function, most of the experimental results are arranged according to {5G, 4G, Wire, WiFi}. But, under the same network conditions, different RL algorithm performance starts to emerge. In WiFi and 4G network environments, the time required for the A2C learning rate to the plateau is higher than that of other networks. And PPO-LSTM learns more slowly than other algorithms in all four network conditions. And PPO-LSTM learns more slowly than other algorithms in all four network conditions. We conjecture that it is due to the gradient problem of RNN in the LSTM network that causes the slow learning rate when dealing with long sequence problems. Using Reward as the evaluation metric, we find that networks with high bandwidth tend to have higher Rewards. There are also exceptions, and we find that PPO-LSTM can stabilize Reward between 4 and 5 under different networks through experiments. Compared with PPO, the advantage of PPO-LSTM is more obvious. DQN is also capable of achieving Reward stability within a certain range, but its Reward is lower than that of PPO-LSTM. Therefore, PPO-LSTM becomes the main RL algorithm used in Nuwa-RL.

C. The performance of Nuwa-RL

PPO-LSTM has an excellent performance in the learning process. We use PPO-LSTM to adjust the k of Nuwa so that the Nuwa can adapt to the changes between different network environments. We use a real network scenario to validate the trained model. The four network types Wire, WiFi, 4G, and 5G are selected. The experiments are performed in the form of file transfers and we use different congestion control methods in a 40-second time frame. Package in flight, One-Way Delay, RTT, and average Throughput are selected as evaluation metrics. Each algorithm experiments several times in different networks and the results are averaged. In the final data presentation, we divide the metrics in the same network environment by the largest metric to compare the gap between the algorithms. The experimental results are presented in Figure 5.

The current algorithm CUBIC built into the Linux kernel and the BBR algorithm proposed by Google is selected as the comparison algorithms in the experiments. Hd-TCP [3] is also selected as the comparison algorithm for the RL algorithm. From the experimental results, we find that the BBR detection-based algorithm is more advantageous in a



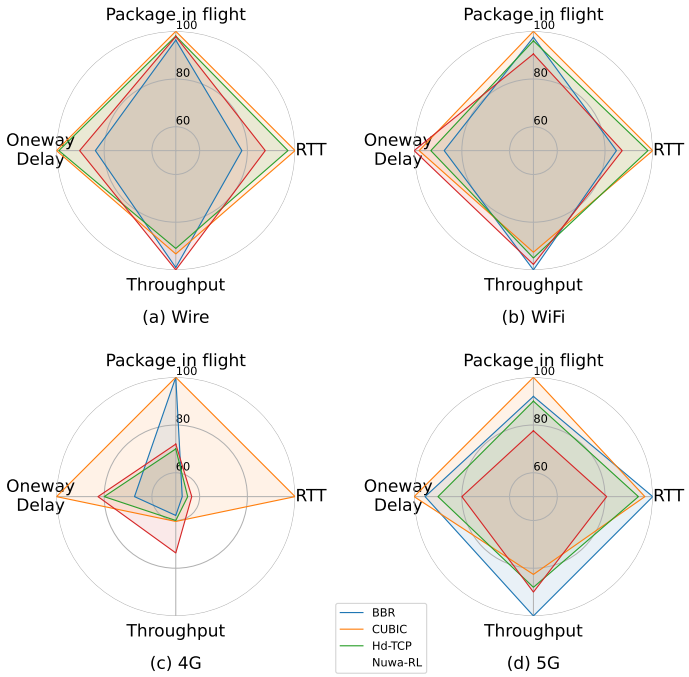
(a) A2C learning rate.

(b) DQN learning rate.

(c) PPO learning rate.

(d) PPO-LSTM learning rate.

Fig. 4. Comparison between different RL algorithms. The better RL algorithm strikes a balance between faster learning efficiency and higher reward values.



(a) Wire

(b) WiFi

(c) 4G

(d) 5G

Fig. 5. Compare the Package in flight, RTT, Throughput, and One-Way delay during data transmission in the same network to verify the protocol's control performance. The values of a particular dimension are divided by the maximum value of that dimension in the figure to facilitate comparison.

stable network transmission environment. It is able to maintain low transmission delay and queueing delay in the network while having high data transmission capability. However, in wireless networks, BBR obviously loses its advantage. In the 4G network environment, the BBR algorithm transmits almost no data during most of the data transmission. CUBIC uses packet loss as a congestion signal in different network environments, causing CUBIC to send too much data to the network. This results where CUBIC's data transmission being accompanied by extremely high data transmission delay while maintaining high throughput.

Among the reinforcement algorithms, the Hd-TCP algorithm is also able to achieve higher throughput based on achieving lower latency, especially in 4G and 5G. In the experiments, the jitter of 4G Trace is greater than that of other Traces. However, Nuwa-RL also achieves the goal of achieving better data transmission capability in multiple networks in data transmission experiments. In wired networks, Nuwa-RL achieves throughput similar to CUBIC while providing better control of queueing delay than CUBIC. When the network environment is switched, especially under wireless networks. Nuwa-RL integrates Nuwa's resilience to wireless network fluctuations. Nuwa-RL is also able to maintain high data transmission capacity in wireless networks.

D. Fairness

In the real network, a good congestion control algorithm not only achieves the occupation of free bandwidth but also shares the bandwidth fairly with other algorithms is one of the evaluation criteria. In Nuwa-RL, its fairness is ensured by (1) the value of k and (2) the use of the alpha-fair function. When the available bandwidth changes, modifying the k value can effectively adjust the magnitude of the window change and achieve the purpose of fast window modification. The application of the alpha-fair function in Reward ensures that the value of Reward is not too high in a multi-stream environment leading to a bandwidth-exclusive situation. Therefore, we conducted fairness experiments on NUwa-RL. We conducted multi-stream transmission experiments in a local simulated experimental environment. We use CUBIC as a control sample since CUBIC is the current default congestion control algo-

rithm for Linux. A new stream is added every 10 seconds to the link of the same bottleneck in our network. Two Nuwa-RL streams and two CUBIC data streams, respectively.

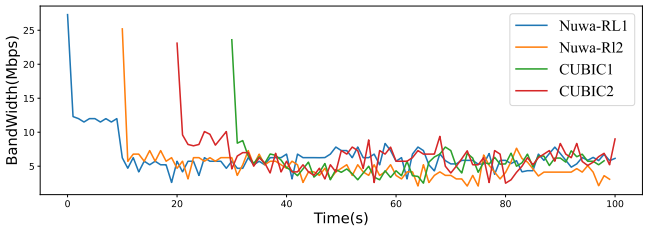


Fig. 6. Fairness experiment. Nuwa-RL is able to share the link bandwidth with other congestion control algorithms when they are on the same link.

The experimental results are shown in Figure 6. After training, Nuwa-RL can effectively cope with multi-stream scenarios during data transmission. When a change in network delay is detected, Nuwa-RL modifies the change direction of the window in a timely manner. And the adjustment of the k-value enables Nuwa-RL to modify the window quickly and drop the window modification to the target level and maintain it in a timely manner. When a packet-drop-based flow joins, Nuwa-RL can adjust the k-value in time to share the bandwidth fairly with the new flow after a slight drop in throughput.

VI. Related Work

A. Traditional TCP Congestion Control

Server-side oriented congestion control protocols: Most of the data in the network environment are transmitted base on TCP, so congestion control of TCP has been one of the research topics of great interest. Congestion control algorithms are critical to the performance of network data transmission. Based on different congestion feedback, server-side congestion control algorithms can be classified into three categories: loss-based algorithms, delay-based algorithms, and delay-based combined loss algorithms. TCP Reno [9], and TCP NewReno [10] were among the early approaches that were loss-based congestion control algorithms. HSTCP [13] and CUBIC [11] modify the window growth model to achieve high network utilization quickly. CUBIC is the default congestion control algorithm in the Linux kernel. Delay-based protocols (e.g., TCPVegas [14]) detect network congestion and adjust cwnd based on RTT, which can react to network conditions faster than packet-drop-based algorithms. However, the delay-based approach suffers significant throughput degradation when competing with loss-based algorithms such as TCP-Reno. In addition, CTCP [12] combines delay-based components into a loss-based TCP congestion avoidance algorithm. TCP BBR [15] estimates the bottleneck bandwidth and RTT latency. It uses distributed control loops to achieve an optimal state that exploits the network while maintaining small queues.

B. RL-based Congestion Control

To address the problem of rule-dependent algorithms that are not adapted to the network environment, learning-driven algorithms have received a lot of attention in recent years. QTCP [16] uses the Q-learning algorithm in the field of RL. It transforms the congestion window adjustment problem into a RL-based problem, thus the algorithm enabling automatic adjustment of the packet sending rate. However, the real network has a large state space making the method extremely robust. Hd-TCP [3] combines heuristic algorithms with deep RL-based algorithms, and uses different policies for different states to improve the robustness of the algorithm. This enables Hd-TCP to achieve effective transmission in actual network transmission.

VII. Conclution

We implemented Nuwa's version based on reinforcement learning, called Nuwa-RL. We use PPO-LSTM to solve the problem that Nuwa cannot adapt to different wireless networks. Through experiments, we provide that PPO-LSTM has better learning performance in wireless network. With PPO-LSTM, we achieve high throughput in different networks while satisfying low latency. Moreover, we use an alpha-fair function combined with Nuwa's original mechanism to achieve fair bandwidth sharing with other congestion control algorithms.

References

[1] Li W, Zhou F, Chowdhury K R, et al. QTCP: Adaptive congestion control with reinforcement learning[J]. IEEE Transactions on Network Science and Engineering, 2018, 6(3): 445-458.

[2] Kong Y, Zang H, Ma X. Improving TCP congestion control with machine intelligence[C]//Proceedings of the 2018 Workshop on Network Meets AI & ML. 2018: 60-66.

[3] Cui L, Yuan Z, Ming Z, et al. Improving the congestion control performance for mobile networks in high-speed railway via deep reinforcement learning[J]. IEEE Transactions on Vehicular Technology, 2020, 69(6): 5864-5875.

[4] Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation[J]. Advances in neural information processing systems, 1999, 12.

[5] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

[6] Srikant R, Başar T. The mathematics of Internet congestion control[M]. Boston: Birkhäuser, 2004.

[7] Andrychowicz O A I M, Baker B, Chociej M, et al. Learning dexterous in-hand manipulation[J]. The International Journal of Robotics Research, 2020, 39(1): 3-20.

[8] Huang, et al., "The 37 Implementation Details of Proximal Policy Optimization", ICLR Blog Track, 2022.

[9] Allman M, Paxson V, Blanton E. TCP congestion control[R]. 2009.

[10] Floyd S, Henderson T, Gurtov A. The NewReno modification to TCP's fast recovery algorithm[R]. 2004.

[11] Ha S, Rhee I, Xu L. CUBIC: a new TCP-friendly high-speed TCP variant[J]. ACM SIGOPS operating systemsreview, 2008, 42(5): 64-74.

[12] Kim M J, Cloud J, ParandehGheibi A, et al. Network coded tcp (ctcp)[J]. arXiv preprint arXiv:1212.2291, 2012.

[13] Chase J S, Gallatin A J, Yocum K G. End system optimizations for high-speed TCP[J]. IEEE Communications Magazine, 2001, 39(4): 68-74.

[14] Brakmo L S, Peterson L L. TCP Vegas: End to end congestion avoidance on a global Internet[J]. IEEE Journal on selectedAreas in communications, 1995, 13(8): 1465-1480.

[15] Cardwell N, Cheng Y, Gunn C S, et al. BBR: congestion-based congestion control[J]. Communications of theACM, 2017, 60(2): 58-66.

[16] Li W, Zhou F, Chowdhury K R, et al. QTCP: Adaptive congestion control with reinforcement learning[J]. IEEE Transactions on Network Science and Engineering, 2018, 6(3): 445-458.