MA-CC: Cross-Layer Congestion Control via Multi-Agent Reinforcement Learning

Jianing Bai[1], Tianhao Zhang[1,2(✉)], Chen Wang[1], and Guangming Xie[1]

[1] Peking University, Beijing 100871, China
tianhao_z@pku.edu.cn
[2] Tsinghua University, Beijing 100084, China

Abstract. Deep reinforcement learning (DRL) injects vigorous vitality into congestion control (CC) to efficiently utilize network capacity for Internet communication applications. Existing methods employ a single DRL-based agent to perform CC under Active Queue Management (AQM) or Transmission Control Protocol (TCP) scheme. To enable AQM and TCP to learn to work cooperatively, this paper aims to study CC from a new perspective from the multi-agent system by leveraging multi-agent reinforcement learning (MARL). To this end, we propose a MARL-based Congestion Control framework, MA-CC, which enables senders and routers to gradually learn cross-layer strategies that dynamically adjust congestion window and packet drop rate. We evaluate the proposed scheme in a typical dumbbell-like network model built on the ns-3 simulator. The results show that MA-CC outperforms traditional rule-based and learning-based congestion control algorithms by providing higher throughput while maintaining low transmission latency.

Keywords: congestion control, multi-agent reinforcement learning

1 Introduction

Recently, successful applications in automatic driving, video streaming, and online games require higher Quality of Service (QoS) for the data transmission environment, which poses new challenges in the design of network protocols in different layers to perform congestion control (CC).

There are two primary schemes devoted to controlling network congestion. One is the transmission control protocol (TCP) CC [1], which is deployed at the transmission layer to avoid congestion by adjusting sending rate. The other is the active queue management (AQM) [2], which is deployed at the network layer to control the buffer to avoid overflow [3]. The traditional TCP/AQM system is rule-based, tuning parameters manually to adapt to network communication environments. However, since the network environment is complicated, it is difficult for designers to obtain expert knowledge about the background to design one rule-based mechanism for various scenarios. Therefore, an intelligent CC scheme is required to cope with the challenges of the complex network environment.

Fortunately, reinforcement learning (RL) has shown massive potential for real-time decision-making under dynamic environments and has been applied to various complex real-world tasks in recent years [4]. As a result, there is an increasing number of researchers who leverage RL on communications and networking [5], especially congestion control, to utilize network capacity efficiently [6]. However, existing learning-based CC protocols mostly use a single agent to perform CC under TCP/TCP scheme by adjusting the congestion window (CWND) or packet drop rate separately [7]. No work enables AQM to cooperate efficiently and intelligently with TCP.

In this paper, we consider congestion control a multi-agent decision-making problem and propose a novel framework called MA-CC, which makes AQM and TCP learn to cooperate using multi-agent reinforcement learning (MARL). Precisely, MA-CC consists of two types of agents: TCP agents that dynamically adjust CWND and AQM agents that control packet drop rate. Using a typical MARL method, the value decomposition network (VDN) algorithm [8], two types of agents can work cooperatively to perform the cross-layer congestion control. The performance of MA-CC is evaluated in a typical dumbbell-like network model built upon the ns-3 simulator. Compared to the existing rule-based and learning-based CC algorithms, MA-CC achieves state-of-the-art performance in terms of throughput and delay. The main contributions of this paper are summarized as follows:

- As far as the authors are aware, it is the first time that a MARL-based approach enabling TCP and AQM to learn to cooperate is proposed to address the cross-layer congestion control problem. Simulation results show that our proposed MA-CC method outperforms typical rule-based and learning-based congestion control algorithms.
- We design a typical dumbbell-like network scenario under the ns3-gym simulator by modeling it as a multi-agent decision-making problem, which speeds up research and development of MARL in the congestion control area.

2 Related Work

In this section, we first introduce some existing traditional rule-based network protocols and then discuss the work that exploits machine learning for networking protocols, primarily focusing on reinforcement learning for congestion control mechanisms.

2.1 Rule-based Protocols

There are various rule-based protocols for solving the congestion control problem, which can be divided into end-to-end CC and network-assisted CC. End-to-end mechanisms, usually applied to TCP CC, rely on implicit signals from the networks, such as delay and the loss of packets. Tahoe [9] and Reno [10] introduce three core phases in CC: slow start, congestion avoidance, and fast recovery. Based on Reno, NewReno [1] is a classical and default congestion control protocol in use today. BBR [11] is a novel mechanism that performs well in

TCP by continuously detecting the maximum link capacity and employing two parameters, namely RTprop and BtlBw, to model the end-to-end network capacity. Besides, Vegas [12], fast active queue management scalable TCP (FAST) [13], low latencies TCP [14], Timely [15] treat increasing RTT as a congestion signal and adjust CWND to keep RTT in the desired range. Moreover, Veno [16], Africa [17] and Google Congestion Control (GCC) [18] combine the loss and delay signals to evaluate congestion.

Although the principle of end-to-end CC is simple to realize, it cannot identify the network environment status precisely only using those implicit signals. For example, packet loss is not necessarily caused by congestion but may also be caused by physical line failure, equipment failure, virus attack, routing information error, etc. To crack this nut, queue length management in network-assisted CC mechanisms that works at the network layer has been proposed. There is a body of research related to the AQM scheme. The RED algorithm [2] is the default algorithm to realize router congestion control, which marks the data packets and drops packets with a certain probability that arrive at the router. The controlling queue delay (CoDel) algorithm [19] is a packet-sojourn time-based AQM, which tracks the minimum queuing delay experienced by the packets. Based on CoDel, the proportional integral controller enhanced (PIE) algorithm [20] improves robustness by using additional parameters.

2.2 Learning-based Protocols

The dynamic and complexity of network scenarios have brought significant challenges for CC. Thus, over the past fifteen years, there has been a lot of effort to implement intelligent congestion control solutions to improve the network system's performance.

In the transfer layer, Remy [21], based on the customized objective function consisting of throughput and delay, attempts to find a mapping from a pre-computed lookup table. Instead of using hardwired mapping, PCC[22] and PCC-Vivace[23] leverage empirically observed performance metrics and online(convex) optimization on machine learning techniques to choose the best sending rate automatically. Moreover, Orca [24] combines the traditional CC algorithm Cubic and RL to compute the CWND. Similarly, RL-TCP [7] uses RL to change the CWND of TCP.

Besides the above mechanisms that adopt the collaboration of senders and receivers, CC mechanisms work at the network layer. For example, QRED [25] adjusts the maximum dropping probability according to the network situation based on the RED scheme and the Q-learning algorithm. RL-AQM [26] also presents a new AQM algorithm based on RL to manage the network resources to keep the low queueing delay and the packet loss rate in different communication situations. Although these learning-based works use RL to inject vigorous vitality into CC, they all consider the problem from the perspective of a single agent, which cannot make TCP and AQM cooperatively perform cross-layer congestion control.

3 MARL-based Congestion Control

In this section, we propose a novel MARL-based framework that uses both TCP agents and AQM agents to dynamically and cooperatively perform cross-layer congestion control, called MA-CC.

3.1 Overview

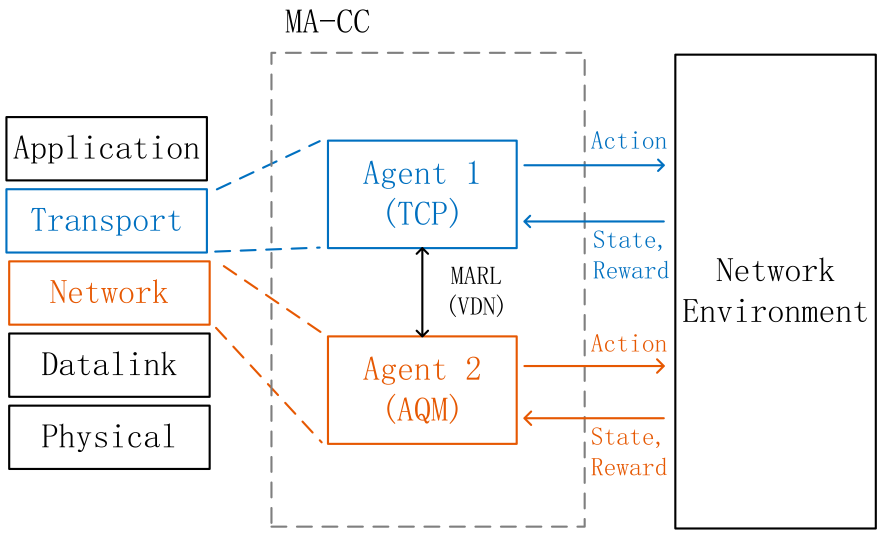

Fig. 1. Architecture of the proposed MA-CC schemes.

The framework of MA-CC is illustrated in Fig. 1. Key to our design is the insight that it is necessary to improve network performance by cooperating AQM scheme with TCP congestion control. Therefore, we integrate a multi-agent reinforcement-based framework with TCP and AQM design in our MA-CC approach to perform cross-layer congestion control cooperatively.

To this end, we consider the cross-layer congestion control as a sequential decision-making problem and formulate it as a decentralized partially observable Markov decision process (Dec-POMDP), defined by a tuple $\langle(\mathcal{N}+\mathcal{M}), \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \mathcal{Z}, \gamma\rangle$. $\mathcal{N}$ is the set of TCP agents with $|\mathcal{N}| = N$, $\mathcal{M}$ is the set of AQM agents with $|\mathcal{M}| = M$, and $\mathcal{S}$ is the set of states. At each time step, each agent $i \in (\mathcal{N}+\mathcal{M})$ chooses an action a_i from its action set $\mathcal{A}_i$, and all agents together take a common action $\mathbf{a}$. The state $s \in \mathcal{S}$ transitions to the next state s' according to the transition function $\mathcal{P}(s'|s, \mathbf{a})$ and all agents receive a shared reward $r(s, \mathbf{a})$. Moreover, each agent only obtains a partial observation

$o_i \in \mathcal{O}_i$ according to the observation function $\mathcal{Z}(s,i) : \mathcal{S} \times (\mathcal{N} + \mathcal{M}) \rightarrow \mathcal{O}_i$ and learns an individual policy $\pi_i(a_i|o_i)$. The objective of all agents is to maximize the cumulative return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\gamma \in [0,1]$ is the discount factor.

According to the above modeling, the MA-CC consists of the following elements:

– Agents: There are two types of agents, TCP agents that work at the transmission layer and AQM agents that work at the network layer.
– State: It consists of the bounded histories of the network statistics and measurements that an agent can obtain from the outside environment.
– Action: Under the related TCP/AQM scheme, the agent chooses an action after perceiving the current state, according to its RL-based policy.
– Reward: It reflects the desirability of the action picked to perform the cross-layer congestion control.

In short, two types of RL-based agents interact with each other and the network environment. They aim to obtain high rewards to improve network communication performance. They take actions (e.g., varying enqueue rate and sending rate) after observing environment states (e.g., transmission delay and throughput). Next, we describe the state, action, and reward of MA-CC in detail.

3.2 Task Description

According to the TCP/AQM scheme, we separately design the state, action, and reward for the TCP agent (labeled as $Agent^1$) and the AQM agent (labeled as $Agent^2$).

STATE: At each time step t, the system monitors the state of the network environment and forms a statistical observation for each agent. The observation of $Agent^1$ is $o_t^1 = (segmentsAcked_t, bytesInFlight_t, RTT_t)$, and the observation of $Agent^2$ is $o_t^2 = (queueLength_t, dequeueRate_t, curr-\ QueueDelay_t)$, which are defined as follows:

– *segmentsAcked*: It is the sum of Segments Acknowledged, indicating the number of segments acknowledged by the receiver in a fixed time, which reflects the available bandwidth and its variation.
– *bytesInFlight*: It is the sum of Bytes in Flight, indicating the number of bytes that have been sent out but unacknowledged by the receiver, which is an essential indicator of optimal Kleinrock's point.
– *RTT*: It is the Round-Trip Time of a packet, indicating the amount of time it takes for a data packet to go from the sender to the receiver and back, which is a key element of the network latency.
– *queueLength*: It is the queue length of packets, indicating the remaining buffer space.
– *dequeueRate*: The dequeue Rate is an important indicator of the packet processing rate and the link's capacity.
– *currQueueDelay*: It is the current queuing delay, which jointly affects the RTT and the response speed of the communication with the propagation delay.

ACTION: In our formulation, the TCP agent ($Agent^1$) is deployed at the transmission layer, whose actions influence the sending rates. That is, at each time step t, $Agent^1$ adjusts the current CWND, i.e., $CWND_t$, to increase, decrease, or maintain it by three discrete actions:

$$CWND_t = \begin{cases} CWND_{t-1} + \mathit{segmentSize}, & a_t^1 = 0 \\ \max(CWND_{t-1} - \mathit{segmentSize}, 1), & a_t^1 = 1 \\ CWND_{t-1}, & a_t^1 = 2 \end{cases} \tag{1}$$

where *segmentSize* indicates the maximum amount of TCP data sent in each segment. As for the $Agent^2$, under the AQM scheme, it executes an action to adjust the buffer queue length. Specifically, at each time step interval t, $Agent^2$ determines the probability p_t of dropping an incoming packet in each flow by two discrete actions:

$$p_t = \begin{cases} 0, \, a_t^2 = 0 \\ 1, \, a_t^2 = 1 \end{cases} \tag{2}$$

where it decides the packet dropping probability happens in each flow during i time interval.

REWARD: The control object is to get high QoS, i.e., to maximize throughput and stability while minimizing delay and packet loss rate. Therefore, according to the performance metrics of TCP and AQM schemes, we design the reward functions for $Agent^1$ and $Agent^2$ as follows:

$$\begin{cases} r_t^1 = & a \times \mathit{segmentsAcked}_t - b \times \mathit{bytesInFlight}_t - c \times RTT_t \\ r_t^2 = & d \times \mathit{dequeueRate}_t - e \times \mathit{currQueueDelay}_t - f \times \mathit{queueLength}_t - g \times \mathit{lossRate}_t \end{cases} \tag{3}$$

where $(a, b, c, d, e, f, g) > 0$ are predetermined constant, r^1 consists of three components to reflect loss rate, throughput, and delay, and r^2 consists of four elements to reflect throughput, delay, loss rate and stability.

3.3 Value Decomposition Network

To demonstrate the capability of the proposed MA-CC, we utilize a typical cooperative MARL method, value decomposition network (VDN) algorithm [8], to learn policies to choose actions and achieve cross-layer congestion control. Specifically, our adopted VDN is a value-based MARL method, which assumes the total Q-value of the multi-agent system. Under the centralized training with decentralized execution (CTDE) paradigm, it can be decomposed into the sum of the Q-values of each agent as follows:

$$Q_{\mathrm{tot}}((o^1, \ldots, o^n), (a^1, \ldots, a^n)) \approx \sum_{i=1}^{n} Q^i(o^i, a^i), \tag{4}$$

where the individual Q-value Q^i of each agent i is updated by minimizing the td-error of Q_{tot}, i.e.,

$$J_{Q_{\mathrm{tot}}} = \mathbb{E}_{(o_t, a_t, r_t, o_{t+1}) \sim D}[Q_{\mathrm{tot}}(\mathbf{o}_t, \mathbf{a}_t) - \hat{Q}_{\mathrm{tot}}(\mathbf{o}_t, \mathbf{a}_t)]^2, \tag{5}$$

Algorithm 1 The VDN method for MA-CC.

```
 1: Randomly initialize Q network $Q^i_{\theta^i}(o^i, a^i)$ with weights $\theta^i$ for each agent $i \in \{1, 2\}$;
    initialize target network $\bar{Q}$ with weights $\bar{\theta}^i$
 2: Initialize the empty replay buffer $D$; initialize the target replace rate $\tau$
 3: for $e = 0$ , $M - 1$ do
 4:   Initialize networking simulation environment
 5:   Receive initial observation $o^1_0$ and $o^2_0$
 6:   for $t = 0$, $T - 1$ do
 7:     For each agent $i$, sample action $a^i$ from $Q^i$ by the epsilon-greedy policy with
        the exploration rate $\epsilon$
 8:     Take the joint observation $\mathbf{o} = [o^1, o^2]$, joint action $\mathbf{a} = [a^1, a^2]$, joint reward
        $\mathbf{r} = [r^1, r^2]$, and the joint next observation $\mathbf{o'} = [o'^1, o'^2]$
 9:     Store $< \mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{o'} >$ in replay buffer $D$
10:     Sample a minibatch $B$ from $D$ to update Q networks by minimizing:
        $\mathbb{E}_B[\sum_i Q^i_{\theta^i}(o^i, a^i) - \gamma \sum_i \max_{a'} \bar{Q}^i_{\bar{\theta}^i}(o'^i, a'^i) - r_{\text{tot}}]^2$
11:     Update the target networks:
        $\bar{\theta}^i \leftarrow \tau\theta^i + (1 - \tau)\bar{\theta}^i$
12:   end for
13: end for
```

where D represents the replay buffer, and $\hat{Q}_{\text{tot}}$ is the td-target of the team Q-value as follows:

$$\hat{Q}_{\text{tot}}(\mathbf{o}_t, \mathbf{a}_t) = r(\mathbf{o}_t, \mathbf{a}_t) + \gamma \max_{a_{t+1}} Q_{\text{tot}}(\mathbf{o}_{t+1}, \mathbf{a}_{t+1}), \tag{6}$$

where $\mathbf{o} \doteq (o^1, \ldots, o^n)$ and $\mathbf{a} \doteq (a^1, \ldots, a^n)$.

The training loops are as follows. Specifically, we randomly initialize the Q networks for $Agent^1$ and $Agent^2$. Besides, the empty replay buffer D is initialized. Then, the training process starts. The networking simulation environment is initialized for every training episode e, and the two types of agents perceive their current observations $\mathbf{o}$. Next, two types of agents choose actions $\mathbf{a}$ from Q networks by the epsilon-greedy policy with the exploration rate ϵ, i.e.,

$$a^i = \begin{cases} \arg\max_a Q^i(o^i, a) & \text{if} \quad random(0,1) \geq \epsilon \\ random(\mathcal{A}_i) & \text{if} \quad random(0,1) < \epsilon \end{cases}, \tag{7}$$

where $\epsilon = 0.995^{|D|}$ and $|D|$ represents the data stored in the replay buffer D. The rewards $\mathbf{r}$ are obtained by executing the actions, and the simulation environment transfers to the next state with the next observations $\mathbf{o'}$. Then, the transition $< \mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{o'} >$ is stored in the buffer D, from which a minibatch B is sampled to update Q networks by minimizing $J_{Q_{\text{tot}}}$. The target networks are used to make the target values change slowly to improve the stability of learning. Such an iteration in each episode loops until the time step k reaches the maximum value T. The pseudocode of the VDN for MA-CC is shown in Algorithm 1.

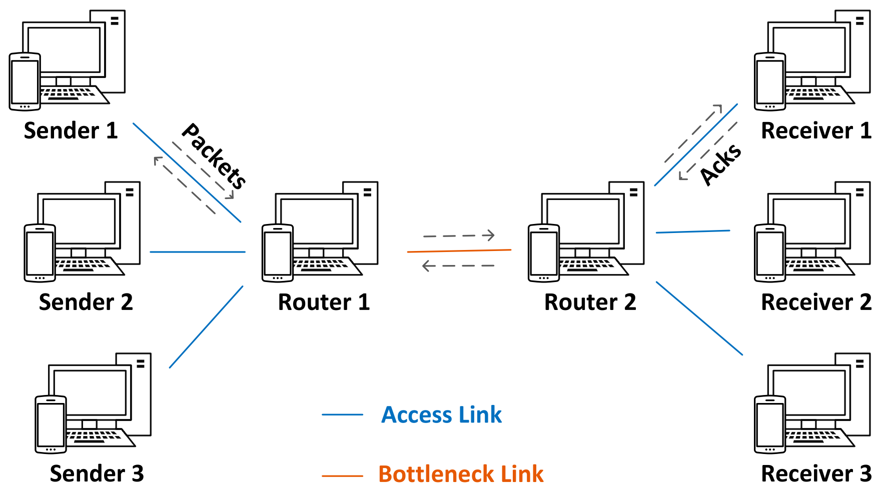

Fig. 2. The dumbbell-shaped network topology with N senders, N routers and O receivers simulation scenarios was implemented in the ns-3 simulator. We set the Bottleneck Link bandwidth as 30Mbps, delay as 100ms, packet error rate as 0.03%, and Access Link bandwidth as 100Mbps, delay as 20ms.

4 Evaluation

In this section, we first give the experimental setup and then elaborate performance of the MA-CC and make comparisons with other rule-based and learning-based congestion control schemes. Finally, we provide a brief analysis of the simulation results.

4.1 Experimental Setup

Recently, the ns-3 simulator and OpenAI Gym are combined to produce a benchmarking simulation system, named ns3-gym [27], for promoting the intersection of reinforcement learning and networking research. Since ns3-gym simplifies feeding the RL with the data from the network system, we build the test scenario and construct experiments in ns3-gym. Specifically, the test network topology built in ns3-gym is indicated in Fig. 2, which represents a natural and complex network environment where multiple flows sent by various devices compete for the Bottleneck link's bandwidth. Furthermore, since we consider congestion control a multi-agent decision-making problem, the TCP and AQM schemes in ns3-gym are modified under MA-CC to connect with MARL.

Table 1. Simulation parameters

Symbol	Parameter Name	Parameter Value
M	Number of TCP agent and sender	3
N	Number of AQM agent and router	2
γ	Discount factor	0.9
α	Learning rate	0.01
τ	Target replace rate	0.05
ϵ	Initial exploration rate	0.9
t_{step}	Step duration	0.2 seconds
-	Queue size	385 packets

4.2 Performance Analysis

Using the VDN algorithm, we trained universal neural networks for each agent and evaluated them online. The discount factor γ is set as 0.9, the initial exploration rate ϵ is set as 0.9, and the target replaces rate τ is set as 0.05. Simulation parameters used in the implementation are summarized in Table 1. Neural network for this task is two fully connected layers with two *ReLU* activation functions. We train the neural network with ten iterations and 1000 episodes for each iteration. To consolidate the proposed MA-CC, we choose well-known rule-based and learning-based protocols at two layers to make a comprehensive comparison, including NewReno, BBR, RED, CoDel, RL-TCP, and RL-AQM.

We consider several critical parameters as our performance metrics, including throughput, delay (RTT), stability of CWND, and queue length. Throughput counts the amount of data successfully transmitted in a given unit of time, measured in Mbps. RTT measures packet transmission delay, which is the amount of time it takes for a signal to be sent, plus the amount of time it takes to acknowledge that signal has been received. The stability of CWND and queue length calculates the mean absolute deviation (MAD) of CWND and queue in a given time interval according to:

$$\overline{d} = \frac{\sum_{i=1}^{n} |x_i - \overline{x}|}{n} \tag{8}$$

The overall simulation results are shown in Fig. 3(A) to Fig. 3(B), where the timeline figures show the performance change of the throughput and delay in the first, fifth, and tenth training episode, respectively. The presence of throughput

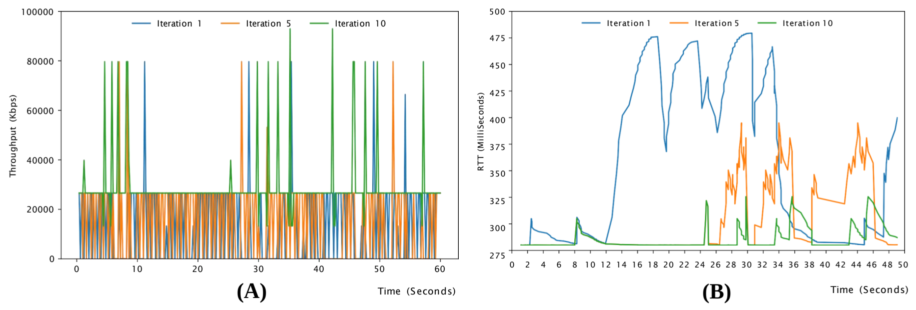

Fig. 3. Real-time (A) throughput and (B) RTT of MA-CC of three iterations.

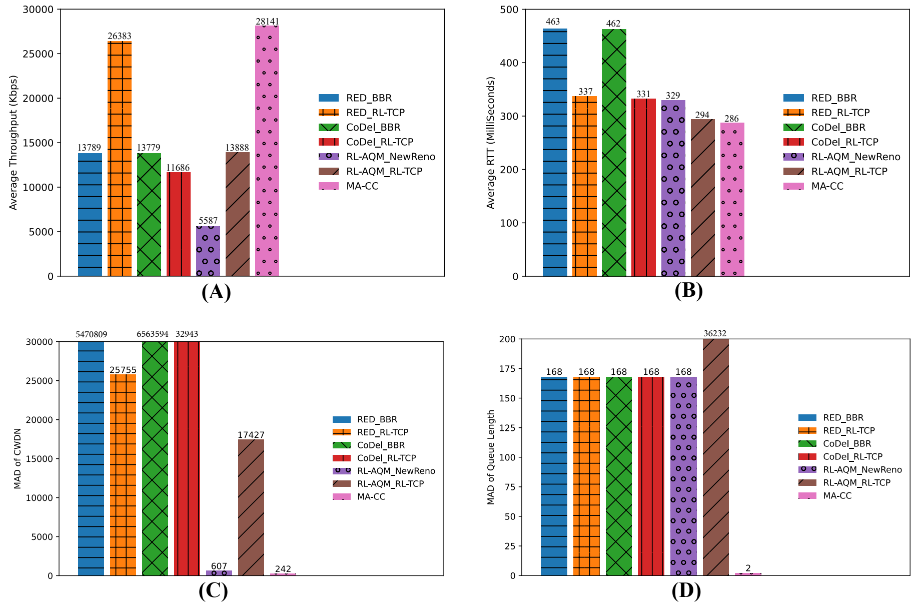

Fig. 4. Comparison results: (A) Average throughput comparisons, (B) Average RTT comparisons, (C) Mean Absolute Deviation of CWND comparisons, and (D) Mean Absolute Deviation of queue length comparisons.

incline and delay decline with training episode increase suggests that two types of agents are gradually learning how to improve performance. MA-CC does not affect congestion control in the first training episode. However, after training several iterations, MA-CC can learn how to improve throughput and reduce delay, and its performance is expected to exceed the traditional CC algorithms. In addition, one can see that the performance of MA-CC tends to be stable after ten training sessions, which indicates the strong learning capability of the MARL-based CC algorithm in the network communication environment.

To further demonstrate the performance of MA-CC, by considering the different trade-offs between latency and throughput of existing schemes, we compare MA-CC with other six AQM and TCP combination protocols, including RED with BBR, CoDel with BBR, RED with TCP-RL, AQM-RL with NewReno, and AQM-RL with TCP-RL. The comparison results are summarized in Fig. 4, including the average throughput, average RTT, mean absolute deviation of CWND, and mean absolute deviation of queue length. From Fig. 4(A), one can see that the average throughput of MA-CC is up to 28459 Kbps while that of the combination of RED and TCP-RL algorithms is up to 26384 Kbps. As for other combinations, their average throughput is lower than 15000 Kbps. From Fig. 4(B), one can see that the average RTT of MA-CC is 287 milliseconds. While that of the combination of AQM-RL and TCP-RL algorithms is similar lower to 294 milliseconds. As for other combinations, their average RTT is higher than 300 milliseconds. Note that the total RTT when empty is $2 * (100 + 2 * 20) = 280ms$. Among these results, one can see that MA-CC achieves both higher throughput and lower latency on average than baselines.

Moreover, from Fig. 4(C), one can see that the MAD of CWND of MA-CC is low to 242 while that of the combination of RL-AQM and NewReno algorithms is 607. As for other combinations, their MAD of CWND is higher than 17000; and from Fig. 4(D), the MAD of queue length of MA-CC is low to 2 while that of the combination of other algorithms is higher than 160 Instead of other schemes that are too fluctuating to remain stable, MA-CC keeps the MAN of CWND and queue length at a relatively low level. This reflects that MA-CC could learn a better control behavior than baselines to maintain more stable performance.

5 Conclusion

In this paper, we novelty formulate a multi-agent decision-making framework to improve congestion control performance and propose an innovative framework with multi-agent reinforcement learning called MA-CC. MA-CC enables AQM and TCP to learn to cooperate at different network layers to configure a suitable CWND and packet drop rate dynamically. By comparing MA-CC with the combination of well-known schemes, our evaluation results in the ns-3 simulator show that with limited training, MA-CC can utilize the network resources efficiently and achieve more stability, higher throughput, and lower latency network communication performance.

In the future, we will implement MA-CC in a real-world communication environment. Besides, we are trying to extend the work of this paper in the aspect of other networking scenarios. We are also interested in exploring new multi-agent reinforcement learning methods suitable for networking and communications based on our proposed MA-CC framework.

References

1. Floyd, S., Henderson, T.T., Gurtov, A.: The NewReno modification to TCP's fast recovery algorithm. RFC 2582 (05 1999)
2. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on networking 1(4), 397–413 (1993)
3. Gettys, J.: Bufferbloat: Dark buffers in the internet. IEEE Internet Computing 15(3), 96–96 (2011)
4. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
5. Chen, Y., Liu, Y., Zeng, M., Saleem, U., Lu, Z., Wen, X., Jin, D., Han, Z., Jiang, T., Li, Y.: Reinforcement learning meets wireless networks: A layering perspective. IEEE Internet of Things Journal 8(1), 85–111 (2021)
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. Computer Networks 192, 108033 (2021)
7. Nie, X., Zhao, Y., Li, Z., Chen, G., Sui, K., Zhang, J., Ye, Z., Pei, D.: Dynamic TCP initial windows and congestion control schemes through reinforcement learning. IEEE Journal on Selected Areas in Communications 37(6), 1231–1247 (2019)
8. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2085–2087 (2018)
9. Jacobson, V.L.: Congestion avoidance and control. ACM SIGCOMM Computer Communication Review (1988)
10. Jacobson, V.: Modified TCP congestion avoidance algorithm. End2end Interest Mailing List (1990)
11. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: BBR: congestion-based congestion control. Communications of the ACM 60(2), 58–66 (2017)
12. Brakmo, L.S., O'Malley, S.W., Peterson, L.L.: TCP Vegas: New techniques for congestion detection and avoidance. In: Proceedings of the conference on Communications architectures, protocols and applications. pp. 24–35 (1994)
13. Jin, C., Wei, D., Low, S.H., Bunn, J., Choe, H.D., Doylle, J.C., Newman, H., Ravot, S., Singh, S., Paganini, F., et al.: FAST TCP: From theory to experiments. IEEE network 19(1), 4–11 (2005)
14. Hock, M., Neumeister, F., Zitterbart, M., Bless, R.: TCP LoLa: Congestion control for low latencies and high throughput. In: 2017 IEEE 42nd Conference on Local Computer Networks (LCN). pp. 215–218. IEEE (2017)
15. Mittal, R., Lam, V.T., Dukkipati, N., Blem, E., Wassel, H., Ghobadi, M., Vahdat, A., Wang, Y., Wetherall, D., Zats, D.: TIMELY: RTT-based congestion control for the datacenter. ACM SIGCOMM Computer Communication Review 45(4), 537–550 (2015)
16. Fu, C.P., Liew, S.C.: TCP Veno: TCP enhancement for transmission over wireless access networks. IEEE Journal on selected areas in communications 21(2), 216–228 (2003)
17. King, R., Baraniuk, R., Riedi, R.: TCP-Africa: An adaptive and fair rapid increase rule for scalable TCP. In: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. vol. 3, pp. 1838–1848. IEEE (2005)

18. Carlucci, G., De Cicco, L., Holmer, S., Mascolo, S.: Analysis and design of the google congestion control for web real-time communication (WebRTC). In: Proceedings of the 7th International Conference on Multimedia Systems. MM-Sys '16, Association for Computing Machinery, New York, NY, USA (2016), https://doi.org/10.1145/2910017.2910605
19. Nichols, K., Jacobson, V.: Controlling queue delay. Communications of the ACM 55(7), 42–50 (2012)
20. Pan, R., Natarajan, P., Baker, F., White, G.: Proportional integral controller enhanced (pie): A lightweight control scheme to address the bufferbloat problem. Tech. rep. (2017)
21. Winstein, K., Balakrishnan, H.: TCP ex machina: Computer-generated congestion control. ACM SIGCOMM Computer Communication Review 43(4), 123–134 (2013)
22. Dong, M., Li, Q., Zarchy, D., Godfrey, P.B., Schapira, M.: PCC: Re-architecting congestion control for consistent high performance. In: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). pp. 395–408 (2015)
23. Dong, M., Meng, T., Zarchy, D., Arslan, E., Gilad, Y., Godfrey, B., Schapira, M.: PCC vivace: Online-learning congestion control. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). pp. 343–356 (2018)
24. 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. pp. 632–647 (2020)
25. Su, Y., Huang, L., Feng, C.: QRED: A q-learning-based active queue management scheme. Journal of Internet Technology 19(4), 1169–1178 (2018)
26. AlWahab, D.A., Gombos, G., Laki, S.: On a deep q-network-based approach for active queue management. In: 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit). pp. 371–376. IEEE (2021)
27. Gawłowicz, P., Zubow, A.: ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research. In: ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) (11 2019)