Computer Networks 192 (2021) 108033

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey paper

When machine learning meets congestion control: A survey and comparison

Huiling Jiang [a,b], Qing Li [b,c,*], Yong Jiang [d,b], GengBiao Shen [d], Richard Sinnott [e], Chen Tian [f], Mingwei Xu [d]

[a] Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, 518055 Shenzhen, China
[b] PCL Research Center of Networks and Communications, Peng Cheng Laboratory, 518055 Shenzhen, China
[c] Institute of Future Networks, Southern University of Science and Technology, 518055 Shenzhen, China
[d] Computer Science and Technology, Tsinghua University, 100091 Beijing, China
[e] School of Computing and Information Systems, University of Melbourne, 3004 Melbourne, Australia
[f] Computer Science, Nanjing University, 210093 Nanjing, China

ARTICLE INFO

Keywords:
Congestion control
Machine learning
Reinforcement learning
Learning-based

ABSTRACT

Machine learning has seen a significant surge and uptake across many diverse applications. The high flexibility, adaptability, and computing capabilities it provides extend traditional approaches used in multiple fields including network operation and management. Numerous surveys have explored machine learning algorithms in the context of networking, such as traffic engineering, performance optimization, and network security. Many machine learning approaches focus on clustering, classification, regression, and reinforcement learning. The innovation of this research, and the contribution of this paper lies in the detailed summary and comparison of learning-based congestion control approaches. Compared with traditional congestion control algorithms which are typically rule-based, capabilities to learn from historical experience are highly desirable. From the literature, it is observed that reinforcement learning is a crucial trend among learning-based congestion control algorithms. In this paper, we explore the performance of reinforcement learning-based congestion control algorithms and present current problems with reinforcement learning-based congestion control algorithms. Moreover, we outline challenges and trends related to learning-based congestion control algorithms.

1. Introduction

As a fundamental component of computer networks, congestion control (CC) plays a significant role in improving the network resource utilization to achieve better performance. With the development of a large number of widely used technologies, e.g., data centers (DCs), WiFi, 5G, and satellite communications, the complexity and diversity of network transmission scenarios and protocols have increased dramatically. This has brought significant challenges to transmission protocol design. A rich variety of CC algorithms have been designed for specific scenarios. However, the variety of network scenarios and more importantly the intrinsic dynamics of the network, make it extremely difficult to design efficient generic CC algorithms. Therefore, CC algorithms based on machine learning (ML) have been proposed to provide a generic CC mechanism that could potentially underpin different network scenarios. In this paper, we provide a background analysis of traditional CC. Based on this, we investigate current works and research challenges in the application of ML in the field of CC.

1.1. Traditional congestion control

The Internet transmission protocol is based on packet switching over best-effort network forwarding [1]. End-to-end transmission control is required to provide a reliable service for applications. To avoid network degradation caused by congestion, CC algorithms are typically employed to improve reliable transmission over the network. Congestion is a state of the network in which it is not capable to deliver the service it was designed for. Network congestion occurs when excessive numbers of data packets are sent over the network by hosts [2]. The objective of CC algorithms is to achieve higher network throughput while avoiding packet loss caused by network overload. CC should ideally also guarantee fairness between end-to-end sessions.

The traditional CC algorithms can be categorized into two types: end-to-end CC [3–5] and network-assisted CC [6–8]. End-to-end approaches only require the collaboration of senders and receivers, and hence they do not rely on any explicit signals from the network. Network-assisted approaches require the support of network devices,

* Corresponding author.
E-mail addresses: jiang-hl19@mails.tsinghua.edu.cn (H. Jiang), liq8@sustech.edu.cn (Q. Li), jiangy@sz.tsinghua.edu.cn (Y. Jiang), sgb16@mails.tsinghua.edu.cn (G. Shen), rsinnott@unimelb.edu.cn (R. Sinnott), tianchen@nju.edu.cn (C. Tian), xumw@tsinghua.edu.cn (M. Xu).

https://doi.org/10.1016/j.comnet.2021.108033
Received 18 November 2020; Received in revised form 17 February 2021; Accepted 16 March 2021
Available online 31 March 2021

e.g. congestion information from routers. These are essential to achieve fairness and responsiveness in complex networking scenarios.

For end-to-end CC, one of the main challenges is to identify network congestion from implicit session signals, including packet loss and transmission delays. There are three main types of end-to-end CC approaches: loss-based CC, delay-based CC, and hybrid CC.

Generally, loss-based approaches such as HighSpeed [9], Hybla [10], and Binary increase congestion control (Bic) [11] adjust the sending rate when a given sender has not received a corresponding acknowledgment (ACK) over a given time, which typically indicates packet loss. Losses are generated when the buffer in a given network device is overloaded, thus loss-based approaches are supposed to attain high throughput by making use of the link bandwidth. However, for some delay-sensitive applications, lower transmission time cannot be guaranteed. Besides, the packet loss may not be triggered by network congestion (e.g., random packet dropping), which may mislead CC decisions.

Therefore, delay-based approaches such as Timely [12] have been proposed. Delay-based approaches rely on detected transmission delays caused by the network. Compared with loss-based approaches, delay-based approaches are more suitable for high-speed and flexible networks such as wireless networks, as they are not influenced by random packet loss. However, calculating the exact transmission delay remains a significant challenge. For example, if there exists a slight change when the packet is processed in the Linux stack, the measured transmission delay will be biased. Thus, the sending rate which is based on the measured delay cannot be controlled precisely.

To take full advantage of both loss signals and delay signals, hybrid approaches such as Veno [13] and an adaptive and fair rapid increase rule for scalable transmission control protocol (Africa TCP) [14] have been put forward. It is noted that these approaches cannot identify the network status precisely based on implicit signals related to packet loss and transmission delay.

To solve this problem, network-assisted CC approaches such as the technique of explicit congestion notification (ECN) [15] have been proposed. The network devices provide explicit signals related to the network status for hosts to make sending rate decisions. When the network device is congested, some packets will be marked with an ECN signal. The receiver will send back the ECN signal in the ACK and the sender will adjust the sending rate accordingly. The ECN signal for congestion is employed in [16]. To further improve CC performance, multi-level ECN signals for congestion are employed in [17], which provides finer-grained CC.

With the dramatically increased complexity and diversity of networks, there are more challenges to conduct CC. Whilst traditional CC approaches may work well in one scenario, they may not guarantee the performance in diverse network scenarios. Furthermore, the changing traffic patterns in one network scenario may also affect the performance of the algorithm. Therefore, an intelligent CC approach is required.

1.2. Learning-based congestion control

The dynamic nature, diversity, and complexity of network scenarios have brought significant challenges for CC. As such, it is difficult to design a generic scheme for all network scenarios. Furthermore, the dynamic nature of even the same network can make the performance of CC unstable. Current network environments may also include both wired networks and wireless networks, making the detection of packet loss more difficult [18–20].

To solve the aforementioned problems, learning-based CC algorithms have been proposed. Different from traditional CC algorithms, learning-based schemes are based on real-time network states to make control decisions instead of using predetermined rules. This allows them to have better adaptability to dynamic and complex network scenarios.

With regards to learning-based CC algorithms, ML techniques are used to train the model including supervised learning techniques, unsupervised learning techniques, and reinforcement learning (RL) techniques. Supervised and unsupervised learning techniques have been widely employed to improve the performance of network CC [18,20,21]. These two techniques are mainly used to estimate the network status, such as congestion signals and queue length. However, these schemes are only partially successful because they are trained offline and are not capable of classifying realistic wireless and congestion loss [19]. While RL has more advantages in dealing with realistic congestion in networks with dynamic and sophisticated state space [22,23]. Therefore, RL techniques are beneficial for CC because of the higher online learning capability [24,25]. At present, much research focuses on RL-based CC schemes.

However, learning-based CC is still in its infancy. Most learning-based CC algorithms adjust the congestion window (CWND) to control the sending rate instead of adjusting the sending rate directly. Therefore, burstiness is still a problem in high-speed networks because the CWND can increase sharply when multiple ACKs arrive [26]. Current learning-based CC algorithms such as [27,28] generally focus on end-to-end CC instead of network-assisted CC. Moreover, current RL-based CC algorithms are not suitable for realistic networks due to time overhead. Thus, designing a general learning-based CC scheme that can work in real network scenarios is still a major goal of both academia and industry.

1.3. Overall analysis

Although learning-based CC algorithms have been a hotspot in research, there is no comprehensive survey on this aspect. There are some related surveys on the combination of ML and network issues. In [29], a survey is conducted to summarize that ML provides amounts of techniques to improve the ability of wireless sensor networks to adapt to the dynamic behavior of network environments. These network issues include scheduling, localization, data aggregation, and so on. This survey provides a detailed summary of the application of ML-based algorithms in wireless sensor networks. Some other surveys on specific network environments related to ML techniques are proposed as well. In [30], a survey is presented to review various ML methods that have been employed for cognitive radios. In [31], the survey gave an overview of the most ML techniques encountered in cellular networks. In particular, there is a comprehensive survey that discussed the common issues, such as routing, CC, resource management, and QoS management when ML techniques are applied [32]. Though this survey covers abundant aspects related to networking, it only gives a rough introduction of CC algorithms. In our survey, we discuss current learning-based CC algorithms and provide systematic analysis and comparison. Moreover, we conduct comprehensive experiments of learning-based CC under dynamic networks and compare them with traditional algorithms. Thereby, this survey of learning-based CC algorithms gives readers exhaustive insight and lays the foundation for subsequent research.

In realistic networks, the implementation of learning-based CC algorithms has shown that they are not efficient as supposed to be because intelligent learning decisions cannot be made fast enough. Therefore, to judge the pros and cons of decision models, we conduct comprehensive experiments of various schemes by using the NS3 emulator [33].

In the simulation, we compare the RL-based CC algorithms of Deep Q Learning (DQL) [34], Proximal Policy Optimization (PPO) [35], and Deep Deterministic Policy Gradient (DDPG) [36] with traditional CC algorithms including NewReno [37], Cubic [38] and Bottleneck Bandwidth and Round-trip Propagation Time (BBR) [39]. We design three different scenarios with different configurations of bandwidth and delay. The network with high bandwidth and low delay simulates a typical data center network. The network with low bandwidth and high delay simulates typical wide area networks. The network with low bandwidth and low delay simulates ad hoc wireless networks (AWNs).

These three network environments represent the diverse environments needed for learning-based CC algorithms.

To fully evaluate the performance of learning-based CC schemes, we generate the DC traffic which includes bulk data transfer which can be called elephant flows, and short-lived data exchange which is also called mice flows. Typical elephant flows are generated by data backup and virtual machine migration while classic mice flows are web browsing and search queries. In DCs, elephant flows account for approximately 80% of total traffics [40]. In our experiments, we generate network traffic traces with 80% elephant flows and 20% mice flows for experiments as well.

The experimental results show that learning-based CC algorithms are more suitable for dynamic environments with higher Bandwidth Delay Product (BDP). For networks with low BDP, i.e. the link bandwidth is low or the link delay is low, learning-based CC algorithms are too aggressive to learn with great stability. Moreover, the performance of these three learning-based CC algorithms shows no difference in our simulated environments because the complexity of the environments is limited. Therefore, all of them can handle these network scenarios.

In realistic scenarios, RL-based CC algorithms are influenced by the computation time needed for RL. This impacts the feasibility of these schemes. Therefore, we propose three potential solutions to deal with this problem. Firstly, design lightweight models based on mapping tables of states and actions to decrease the time consumption of learning decisions. Secondly, decrease the frequency of decisions to provide better feasibility under low-dynamic network scenarios. Finally, asynchronous RL can improve the speed of the decision-making process in RL-based CC algorithms.

Based on this analysis, we further explore the challenges and trends for future works in the area of learning-based CC. Current challenges of learning-based CC algorithms are mainly focused on engineering-related issues such as parameter selection, high computational complexity, high memory consumption, low training efficiency, hard convergence, incompatibility and fairness. Based on the understanding and analysis of the current learning-based CC solutions, we identify trends in learning-based CC. Firstly, because of their capability for dealing with network congestion with dynamic and sophisticated state spaces, RL-based CC will be a significant research trend moving forward. Secondly, the programmable switches can be used in learning-based CC algorithms to give end hosts better insights into network status. Third, given the excessive time and cost of learning decisions, lightweight learning-based CC will be a key research direction. Finally, an open platform that provides massively differentiated dynamic network scenarios to support the exploration and verification of learning-based CC mechanisms, is supposed to be designed.

The rest of the paper is structured as follows. In Section 2, we present related background knowledge. In Sections 3–5, we consider supervised learning-based CC algorithms, unsupervised learning-based CC algorithms, and RL-based CC algorithms respectively as representatives of three main groups of learning-based CC algorithms. In Section 6, we provide an overview of the setup of simulations. In Section 7, we conduct simulations and compare performances between RL-based CC algorithms and traditional CC algorithms. In Section 8, we outline the challenges and trends of learning-based TCP. Finally, in Section 9, we conclude the paper.

2. Background

In this section, we will introduce relevant domain knowledge in the field of CC including the mechanism of CC algorithms, classic CC algorithms, and the performance metrics.

2.1. Congestion control mechanisms

CC mechanisms typically involve four key parts: slow start, congestion avoidance, retransmission, and fast recovery [41]. To illustrate the procedure of CC, we adopt the window-based CC.

Slow start. In the classic slow start process, each time a good ACK is received, it means that the sender can send twice the numbers of packets last sent, which will cause the sender's window to grow exponentially over time.

Congestion avoidance. In the slow start phase, the CWND can grow rapidly, to a given threshold. Once the threshold is reached, it means that only limited network resources are available. Once the slow start threshold is established, TCP will enter the congestion avoidance phase, and increase the value of CWND each time based on the size of the successfully transmitted packets. The increasing speed is much slower than the exponential growth rate during the slow start.

Retransmission. Retransmission includes timeout retransmission and fast retransmission. Timeout retransmission starts a timer after sending a given packet. If no acknowledged packet of the datagram is sent within a certain time, the data is retransmitted until the transmission is successful. Fast retransmission requires the receiver to send a duplicate ACK immediately after receiving an out-of-sequence segment so that the sender knows as soon as possible that a segment has not reached the designated server, rather than waiting to send data confirmation. The retransmission mechanism in CC ensures that data can be transmitted from the sender to the receiver.

Fast recovery. Fast recovery means that when the sender receives three duplicate ACKs in succession, it executes a multiplication reduction algorithm and halves the slow start threshold to prevent network congestion. The CWND increases in an accumulative manner, causing the CWND to increase slowly and linearly. The fast recovery algorithm can avoid congestion and make use of network resources efficiently.

Among traditional CC algorithms, the above four mechanisms make up the basic approaches while learning-based CC algorithms do not adopt strict rules to control congestion. To guarantee the flexibility for different scenarios, learning-based CC algorithms can learn different strategies to adjust the CWND instead of following fixed rules.

2.2. Classic congestion control algorithms

As mentioned before, traditional CC algorithms include end-to-end CC algorithms and network-assisted CC algorithms. In this section, we will discuss some representatives among these algorithms.

2.2.1. End-to-end congestion control algorithms

Among end-to-end CC algorithms, loss-based CC algorithms, delay-based algorithms, and hybrid CC algorithms show different strengths and weaknesses. The summary of these algorithms is presented in Tables 1, 2, and 3 respectively.

Loss-based CC algorithms are early versions of CC techniques and Tahoe [42] is the earliest version of TCP. The main architecture of Tahoe includes Slow Start, Congestion Avoidance, and Fast Retransmit. On the basis of Tahoe, Reno [43] adds a fast recovery mechanism. When three repeated ACKs are received or the Recovery Time Objective (RTO) has expired and an ACK for a certain data packet has not been received, the fast recovery mechanism will consider the packet to be lost and determine that there is congestion in the network. The fast recovery algorithm proposed by Reno improves the throughput and robustness after packet loss, but the drawback is that it only considers the situation where only one packet is lost. As long as one packet is lost, it is considered congestion. Therefore, NewReno [3] is proposed to deal with this problem, which mainly improves the fast recovery mechanism. In the Neweno algorithm, only when all the lost packets are retransmitted and received confirmation will the sender exit. Although NewReno can solve the problem of a large number of packet losses, NewReno can only have one packet loss error per RTT time. In order to deal with the loss of a large number of data packets more effectively, a selectvie ACK (SACK) algorithm [44] is proposed. SACK adds mechanisms of selective ACKs and selective retransmission to Reno. Therefore, the sender can know which data has been received

Table 1
Loss-based CC algorithms.

Algorithm	Details of the algorithm	Improved performance	Limitations
Tahoe [42]	Define the basic structure of CC algorithms including slow start, congestion avoidance, and fast retransmit.	Fast network resource discovery and high efficiency.	Strain the network with high-amplitude periodic phases.
Reno [43]	Add a fast recovery mechanism.	Optimize the slow start mechanism.	Suffer performance degradation when the consecutive packet losses and random losses exist.
NewReno [3]	Modify the fast recovery algorithm by incorporating a response to partial ACKs.	Reduce packet losses and delay if multiple packet losses occur simultaneously.	Limit the throughput due to the prolonged recovery mechanism.
SACK [44]	More information is added in the feedback messages by increasing mechanisms of selective ACKs and selective Retransmission.	More information provided in ACKs.	The length of the option of TCP header is limited and thus the algorithm is not universal.
Westwood [45]	Estimate bandwidth as the feedback of losses information, which satisfies wireless scenarios where losses are random or sporadic.	Throughput and fairness in wireless networks.	Unfair if the network has limited buffering capacity because the estimate of bandwidth gives inaccurate results.
Bic [11]	Combine additive increase scheme and binary search increase scheme, which satisfies different sizes of CWND respectively.	Scalability and TCP-friendliness.	RTT-fairness cannot be guaranteed.
Cubic [38]	Modify the linear window growth function to adjust to the fast and long-distance network environment.	Scalability, stability, and fairness.	More packet losses are produced.
HighSpeed [9]	Modify the TCP response function to adapt the scenarios with large CWND in TCP connections.	Throughput in high-speed long-delay networks.	Cannot guarantee fairness if flows have different RTTs.
Hybla [10]	Adopt the SACK option, timestamps, and packet pacing to reduce the impact of losses, inappropriate timeouts and burstiness.	Fairness and friendliness in long-delay networks.	Unable to work effectively in high-speed networks with a relatively small delay.

Table 2
Delay-based CC algorithms.

Algorithm	Details of the algorithm	Improved performance	Limitations
Vegas [4]	Modify the slow start mechanism to control the number of extra buffers in the network.	Convergence speed and higher throughput.	Suffer low link utilization ratio compared with aggressive algorithms such as Cubic.
LoLa [46]	Present a fair flow balancing mechanism for high-speed wide-area networks.	Utilization, throughput, and fairness.	Cannot coexist with loss-based CC algorithms fairly.
FAST [47]	Present an equation-based algorithm and use queuing delay as feedback to check congestion.	Proportional fairness and not penalize flows with large RTTs.	Available network resources cannot be allocated fairly.
Timely [12]	Use timestamps and fast ACK based on precise RTT measurements to control congestion.	Throughput and latency.	The performance depends on the hardware technology of the network interface card
LEDBAT [48]	Monitor queuing delay and consider the fluctuation of the delay as an early signal of congestion.	Queuing delay for latency-sensitive applications.	Suffer from issues related to incorrect propagation delay estimation such as unfairness and latecomer advantage problems.
Copa [49]	Use the observed evolution of one-way delay to evaluate the target sending rate, and then increase or decrease the CWND depending on whether the current rate is lower or higher than the target rate.	Lower delay without harming throughput and robust to non-congestive loss and large bottleneck buffers.	Suffer from low throughput when competing with loss-based CC algorithms.

by the receiver. In summary, these four algorithms represent the early development of loss-based CC algorithms.

However, in some specific network scenarios, optimized CC algorithms are required. In high-speed networks, basic CC algorithms are not efficient to make use of the bandwidth resources. Some algorithms are designed to deal with this issue. Bic [11] is an example. Bic uses the idea of binary search. When a packet loss occurs, indicating that the optimal window value should be smaller than this value, then Bic sets the CWND at this time to max_win, and the value after the multiplication is reduced to min_win. The optimal value of the CWND is expected to be found between max_win and min_win. But Bic harms fairness and Cubic [38] is proposed to optimize Bic. Cubic uses a cubic function to replace the growth function of Bic. Also, the most critical point in Cubic is that the growth function of CWND only depends on the time interval between two consecutive congestion events, so the growth rate of the CWND is completely independent of the network RTT. Thus, fairness is ensured.

HighSpeed [9] and Hybla [10] are suitable for high-speed networks as well. HighSpeed modifies the reaction function of the standard TCP protocol, which is affected by the combined effect of the growth

Table 3
Hybrid-based CC algorithms.

Algorithm	Details of the algorithm	Improved performance	Limitations
Veno [13]	Monitor congestion level based on RTT and packet losses and determine whether congestion occurs for the networks with random packet loss.	Throughput.	Not address the issue of bursty packet loss in wireless networks.
Africa [14]	Use delay information as an indicator towards an adaptive fair rapid increase for links with high BDP.	Throughput and fairness of high-BDP networks.	Not been implemented in real networks.
Compound [50]	Propose a compound TCP, which adds a delay-based component into Reno algorithm for high-speed and long distant networks.	Scalability and fairness in high-speed and long-delay networks.	Suffer fairness and latecomer advantage issues when base RTT is wrongly estimated.
Libra [51]	Optimize the utility function, which is independent of RTT, to balance TCP fairness and stability.	Stability and fairness.	Not guarantee the performance due to the estimation biases.
GCC [52]	Predict the bandwidth at the receiving end based on the input bit rate and adjust the rate according to the congestion of the link, and the congestion of the link is reflected by the change in the increment of the arrival time interval.	Reduce network delay and smooth data transmission.	Difficult to deploy due to the complex framework.
Remy [53]	Present a computer-generated algorithm, which makes control rules for independent endpoints.	Throughput and delay.	The performance depends on the training data set.
BBR [39]	Detect instantaneous link available maximum bandwidth and minimum RTT.	Throughput and delay in networks with shallow buffers and random losses.	Perform poorly when competing with other CC algorithms such as Cubic because Cubic tends to fill up buffers.
SCP [54]	Send real data packets in the data channel and detect the path condition by sending a stand-in packet in the control channel.	Zero packet loss, full utilization, and zero buffer occupancy are achieved simultaneously.	Not suitable for mice flows and light traffic loads because the overall duration of the concurrent flows is too short for the rate adjustment algorithm.
PCC [55]	Send a rate for a short time, and observe the results such as loss and latency, and calculate the value of utility which is used to update the latter sending rate.	Fairness, stability and stability-reactiveness trade-off compared with TCP such as Cubic.	The delay cannot be optimized well and the convergence speed is too slow.
PCC Vivace [56]	Based on PCC, a gradient-based no-regret online optimization algorithm is used to adjust the sending rate.	Convergence speed and the speed of response in dynamic networks.	Limited flexibility.
PCC Proteus [57]	Extend the utility function of PCC Vivace with three types: scavenger, primary and hybrid.	Flexibility and robustness.	The switching mechanism of utility functions is very rough.

function and the packet loss reduction function. As for Hybla, it makes use of the SACK option, timestamps, and packet pacing to reduce losses and improve throughput.

In order to improve the throughput in wireless networks, Westwood [45] estimates the available bandwidth of the network based on Reno and makes appropriate adjustments to achieve a faster recovery. However, Westwood cannot distinguish between wireless packet loss and congestion loss. While the measurement of delay can also be used to infer the network congestion, which can compensate for the impact of the random packet loss. Thus, delay-based CC algorithms are presented.

Delay-based CC algorithms mainly include Vegas [4], Fast active queue management scalable TCP (FAST) [47], Low latencies TCP (LoLa) [46], and Timely [12].

The main topic of delay-based TCP is to optimize the measurement of delay such as RTT, the one-way delay, and queue delay to obtain a more fine-grained model, which can fully represent the congestion in the networks. For instance, Vegas records the system clock when a segment is sent. Then read the clock again when an ACK arrives and calculate RTT based on the timestamp recorded [4]. However, the fairness of Vegas is hard to achieve. LoLa [46] is an enhanced Vegas that is based on the growth function of Cubic by using a fair flow balancing mechanism. Though LoLa has better inter-flow RTT fairness and better RTT estimation, it cannot coexist fairly with loss-based CC algorithms due to the aggressiveness of loss-based CC algorithms.

FAST is another [47] optimized Vegas, which uses queuing delay as a congestion signal. Specifically, FAST estimates the number of packets in queues by measuring the difference between the observed RTT and the base RTT. Thus, FAST makes larger steps when the system has not reached equilibrium and smaller steps near equilibrium. While Vegas makes fixed size adjustments to the rate. As a result, the policy of rate adjustments is independent of how far the current rate is from the target rate. Therefore, the speed of convergence based on FAST is faster compared with Vegas.

However, in DCs, the RTT is relatively small, and software timestamping is too inaccurate to measure RTT. Thus, hardware timestamping is provided by modern Network Interface Cards (NICs) to obtain more precise microsecond RTT measurements, which is applied in Timely [12].

Instead of measuring RTT, Low Extra Delay Background Transport (LEDBAT) [48] adopts a one-way delay to measure the congestion of networks. The advantage of the one-way delay over RTT is that it does not need to consider the loop delay experienced when ack returns. Based on LEDBAT, when there is no other flow in the bottleneck link, the flow can occupy the full bandwidth and make full use of the network resources. At the same time, LEDBAT can ensure low queuing delay. When Cubic competes for bandwidth, LEDBAT actively grants bandwidth to reduce the sending rate. Therefore, LEDBAT is very friendly to Cubic flows.

Copa [49] uses the one-way delay to measure the congestion as well. In Copa, not only low latency is achieved, but the balance of throughput and latency can also be configured. The larger the parameter, the more sensitive to latency. Copa uses the competitive mode to solve the

same radical problem. Once it detects abnormal traffic, it will make adjustments to make itself less and less aggressive. Therefore, when used with QUIC, Copa provides higher quality and lower latency for mobile live broadcast than Cubic and BBR. Moreover, Copa has lower transmission RTT and lower loss overhead.

Hybrid CC algorithms use loss and delay to jointly evaluate congestion. In this way, High throughput and low latency may be achieved at the same time. There are some typical hybrid CC algorithms such as Veno [13], Africa [14], Compound [50], Libra [51], and Google Congestion Control (GCC) [52], the first four of which are based on Reno.

Veno combines Vegas and Reno. Vegas can measure the number of data packets belonging to this connection in the network bottleneck router. Veno uses this variable to distinguish random packet loss from congestion packet loss and takes different actions. Veno also improves the growth function of CWND that is the basic part of Reno. When the number of packets belonging to this connection in the network bottleneck router exceeds a certain value, the growth rate of CWND will be slowed down and thus packet losses will be reduced. Experiments show that Veno can improve throughput significantly without adversely affecting other concurrent TCP connections compared with Vegas and Reno [13].

Africa [14] is based on Reno as well, which can switch modes according to network congestion. When bandwidth resources are sufficient, Africa uses an aggressive, scalable window increase mechanism. RTT is used to measure congestion as well. If the congestion level shows that bandwidth resources are highly utilized, Africa will adopt Reno's congestion avoidance mechanism. Therefore, Africa not only achieves high throughput but also ensures fairness.

Compound [50] is also a hybrid algorithm used to solve the problem of poor RTT fairness among loss-based CC algorithms. Compound maintains two CWNDs, one is a standard window similar to the Reno mechanism and the second is a scalable delay window based on Vegas. The CWND is calculated based on the summation of these two windows. Thus, Compound combines a scalable delay-based component with a standard loss-based component. When the network is congested, the delay-based component will reduce the sending rate significantly. But the throughput will be lower bounded by Reno. Therefore, there is limited RTT unfairness due to delay-based schemes, and the throughput is guaranteed because of the loss-based schemes. As for the issue of RTT unfairness, Libra [51] uses non-linear optimization to guarantee fairness among TCP flows regardless of RTT.

Different from the above algorithms, GCC [52] is applied in WebRTC which uses UDP-based RTP to transmit media data instead of TCP. In GCC, the rate control based on the packet loss rate runs on the sender and relies on RTCP RR messages to work. WebRTC receives the RTCP RR message from the receiver at the sender and dynamically adjusts the code rate at the sender according to the packet loss rate information carried in its Report Block. Delay-based rate control runs on the receiving end.

Besides, there are some innovative hybrid algorithms that have received extensive attention including Remy [53], BBR [39], Stand-in Control Protocol (SCP) [54], Performance-oriented Congestion Control (PCC) [55], PCC Vivace [56], and PCC Proteus [57].

In Remy [53], the utility function consists of throughput and delay. To maximize the expected value of the utility function, Remy finds the mapping based on a pre-computed lookup table. Thence, the corresponding sending rate is estimated. To converge to the optimal sending rate and fully utilize the network, BBR estimates the available bandwidth based on the probing mechanism [39], which is representative of adjusting the sending rate by detecting the network status. Learning from the probing mechanism of BBR, researchers proposed Stand-in Control Protocol (SCP) in [54]. To achieve fast convergence, SCP partitions a physical link into control and data channels. Before determining the actual sending rate in the data channel, SCP sends stand-in packets in the control channel to probe the network condition. Based on the feedback from the control channel, thus data channel is possible to achieve full network utilization. Similar to BBR, the PCC algorithm family is also a method of dynamically adjusting the rate by detecting available network resources including PCC [55], PCC Vivace [56], and PCC Proteus [57]. They show great performance based on carefully designed utility functions that cover basic performance metrics such as round-trip time (RTT). Compared to BBR, PCC converges slower because of PCC's conservative increasing mechanism.

2.2.2. Network-assisted congestion control algorithms

Network-assisted CC algorithms require the coordination of the sender and receiver, and require routers to perform specific processing on packets. Typical algorithms consist of ECN-based CC algorithms [58] and Quantized Congestion Notification-based (QCN-based) algorithms [59]. The summary of network-assisted CC algorithms is presented in Table 4.

ECN is an extension to the Internet Protocol and Transmission Control Protocol. ECN allows end-to-end notification of CC to avoid packet loss and requires specific support from the Internet layer and the transport layer. Generally speaking, TCP/Internet Protocol (IP) networks indicate channel congestion by dropping data packets. In the case of successful ECN negotiation, the ECN-aware router can set a mark in the IP header instead of discarding the packet to indicate that congestion is about to occur. The receiving end of the data packet responds to the sending end's indication, reducing its transmission rate as if it had detected packet loss in the usual way. There are many ECN-based CC algorithms such as DCTCP [26] and High-bandwidth Ultra-Low Latency (HULL) [60]. The advantage of ECN is that it can reduce the number of discarded data packets in a TCP connection to avoid retransmission and reduce waiting time, especially network jitter.

QCN is a set of end-to-end congestion notification mechanisms applied to L2. Through an active reverse notification, the packet loss rate and delay in the network are reduced, thereby improving network performance. QCN includes two parts: Congestion Point (CP) and Reaction Point (RP). CP means that the congested network switching equipment samples the data frames that are being sent in the sending buffer. If congestion occurs, it will generate a Congestion Notification Message (CNM) to the RP of the sampled data frame. When the RP receives the CNM information, it will limit the sending rate of the corresponding message. At the same time, the RP will slowly increase its sending rate to detect the available bandwidth and recover the rate due to congestion. In [61], DCQCN is designed based on QCN and DCTCP. The proposed algorithm dramatically improves the throughput and fairness of the traffic. Though QCN is efficient in controlling the queue length and assists the CC, QCN cannot be deployed and used in large-scale IP networks since QCN cannot be used in IP routing networks.

2.3. Performance metrics of congestion control algorithms

CC algorithms are expected to achieve various goals and objectives as shown in Table 5.

Throughput represents the amount of data that passes through a network (or channel, interface) in a given time interval. High throughput means high link utilization. Maximizing throughput is crucial. Given the link bandwidth, high throughput indicates high efficiency in transferring data. For instance, Bic is aimed to achieve higher throughput based on a more aggressive mechanism compared with Reno [11].

RTT measures the time including the transmission time, the propagation time, the queue time, and the processing time. Flow completion time (FCT) indicates the time required to transfer the flows. RTT and FCT are expected to be small. For users, RTT and FCT show the delays that they may have to tolerate. However, it may be the case that maximizing throughput and minimizing RTT or FCT can be orthogonal. High throughput means making use of the link bandwidth as much as possible, which can give rise to an increased queue length that

Table 4
Network-assisted CC algorithms.

Algorithm	Details of the algorithm	Improved performance	Limitations
ECN [58]	Allow end-to-end notification of CC to avoid packet loss, and require specific support from the Internet layer and the transport layer.	Avoid retransmission and reduce waiting time, especially network jitter.	Performance depends on the accuracy of the AQM used.
QCN [59]	Include CP which is used to generate congestion notification message and send to RP and RP is responsible to deal with it and controls the sending rate.	Ensure low queue buildup and low queue oscillations.	Cannot be deployed and used in large-scale IP networks since QCN cannot be used in IP routing networks.

Table 5
Objectives of learning-based CC algorithms.

Objective	Description
Maximizing throughput [11]	To maximize throughput, bandwidth utilization is supposed to be high. High throughput contradicts low RTT or flow completion time since high throughput means the environment tolerates high queue lengths, which may cause long delays.
Minimizing RTT or flow completion time [4]	Minimizing RTT or flow completion time is a basic requirement expected to be met. For each task, the flow completion time reflects the delay, which is supposed to be as small as possible.
Minimizing packet loss rate [54]	Minimizing the packet loss rate is a basic goal of CC algorithms. Low packet loss rate means that there. is a stable network environment and low delay.
Fairness [9,10]	Fairness is important for multiple users. Resource allocation should be as fair as possible between users and consider diverse applications.
Responsiveness [56]	Updating the frequency and adjustment policy of CWND can influence the responsiveness of algorithms. High responsiveness is expected, which implies high resource-consumption as well. Therefore, responsiveness needs to be balanced based on different scenarios.

may cause delays. In Vegas [4], the crucial insight is to predict the congestion level based on measured RTT. Moreover, the lower RTT and FCT can be realized in Vegas due to the delay-based mechanism.

The packet loss rate indicates the efficiency of the data transmission. For CC, minimizing the packet loss rate is important as it shows the control capability and stability of the network. SCP [54] is an example. By designing two channels, zero packet loss is possible.

Fairness is a measure of equality of the resource allocation of the network. Increased fairness requires CC algorithms to fairly allocate resources between flows to user satisfaction and in turn improve the Quality of Service (QoS). Many algorithms regard fairness as one of the crucial performance metrics like HighSpeed [9] and Hybla [10].

Responsiveness reflects the speed of the CC to deal with real-time flows. A high responsiveness level means that the algorithms can detect the congestion quickly and rapidly adjust the CWND to an optimal value. Compared with PCC [55], the kernel improvement of PCC Vivace [56] is the improved speed of response.

These objectives are important for all CC algorithms, but they are hard to achieve simultaneously. In different scenarios, the targets may also have different priorities, and hence trade-offs are necessary. Based on the previous literature, different CC research focuses on different performance metrics including throughput, RTT, the packet loss rate, and fairness. In our simulations, we measure these four parameters in detail.

3. Supervised learning-based congestion control algorithms

In this section, we introduce supervised learning-based CC algorithms. Supervised learning techniques train given samples to obtain an optimal model, and then use this model to map all inputs to corresponding outputs. By performing judgments on the outputs, supervised learning techniques have the ability to perform data classification. Classic supervised learning methods include decision trees, random forests, Bayes, regression, and neural networks.

In the network field, supervised learning methods are used to implement many network functions, such as traffic classification, CC, resource management, and network security. Among these functions, because optimal traffic classification policies can promote the reasonable allocation of network resources and reduce the probability of network congestion, traffic classification occupies a very significant position and has an important impact on a wide range of network operation and management activities including CC. Therefore, traffic classification based on supervised learning can also be regarded as the application of supervised learning algorithms in the field of CC.

There is a huge amount of supervised learning techniques used for traffic classification. In [62], packet-level information and flow-level information are used as states. Nearest Neighbor and Linear Discriminant Analysis are employed to classify flows into four classes: the interactive class, the bulk data transfer class, the streaming class, and the transactional class. Based on the high-precision classifier, IP network operators are able to provide differentiated QoS for various applications. However, the error rate is up to 9.4%, which is relatively high. In [63], a Naive Bayes Estimator is applied to categorize traffic by application, which improves the accuracy significantly. In this paper, the accuracy is about 95%. The insight provided in this research is that the algorithm proposed is sensitive to its initial assumptions. Moreover, breaking the Gaussian assumptions and improving the quality of discriminators as input led to significant improvements in the accuracy. In [64], more features are taken into consideration such as the size and the number of "burst" packets. The burst feature is supposed to provide fantastic discriminant power among distinctive classes. To reduce the computation cost, a Support Vector Machine (SVM) is proposed [65]. In the presented model, multiple binary SVM classifiers are organized into a tournament structure, which can dramatically decrease the number of training samples. Moreover, the feature selection and parameter choice for individual SVM are different. These optimizations can further improve accuracy and reduce computational cost. The simulation shows that the designed algorithm can reduce computation cost by up to 7.65 times.

To sum up, the extensive application of supervised learning models in traffic classification plays an important role in crucial engineering issues such as classification accuracy and computation cost. Except for traffic classification, supervised learning methods can be used to predict congestion signals for end-to-end networks and manage queue

length for network-assisted networks, which affect the performance of CC directly.

Congestion signal prediction consists of loss classification and delay prediction. As mentioned before, congestion is detected implicitly based on packet loss or delay when congestion occurs in traditional CC algorithms. In supervised learning-based CC algorithms, congestion is estimated in advance based on current and previous network states such as the packet arrival interval and the network delay. The key basis for this approach is that network states form a continuous time series, where the future state can be predicted by past states. As for queue length management, supervised learning methods play an important role in the accurate and efficient prediction of queue length. In the next two parts, a more detailed description will be shown.

3.1. Congestion detection in end-to-end networks

3.1.1. Loss classification

Loss is a crucial but indirect signal used to detect congestion. It gives nodes feedback in networks only when congestion has already happened. In addition, basic loss-based CC algorithms cannot distinguish the cause of packet loss. Therefore, the classification of loss is essential to understand CC.

Wireless networks provide many classic scenarios required to distinguish the wireless loss and congestion loss. In wireless networks, losses may be caused by erroneous wireless links, user mobility, channel conditions, and interference. There has been a body of research related to loss classifications in wireless networks based on traditional CC algorithms. In [66], the proposed algorithm (Biaz) uses the packet inter-arrival time to classify wireless loss and congestion loss. If the packet inter-arrival time is confined to a range, the missing packets are lost due to wireless loss. Otherwise, the loss is considered a congestion loss. In [67], Spike, a newly designed loss classifier for relative one-way trip time (ROTT) was used to differentiate loss types. If the connection of ROTT was relatively higher, the loss was supposed to be caused by congestion. In other cases, the loss was assumed to be a wireless loss. In [68], the number of losses and ROTT were used to distinguish the types of losses. The presented algorithm called ZigZag is more efficient than the above two algorithms.

These loss classifiers are effective in some specific scenarios but have their limitations. Biaz [66] is suitable for wireless last hop topology instead of the wireless bottleneck links with competitive flows while Spike [67] shows better performance in wireless backbone topology with multiple flows. ZigZag [68] is relatively more general, and hence is able to satisfy different topology scenarios but it is sensitive to the sending rate.

Considering the limitations of traditional loss classifiers for wireless networks, supervised learning techniques offer several advantages. To fully understand the loss information, multiple parameters can be taken into consideration. In [69], the multi-layer perceptron technique is used as the classifier. The minimum RTT value and the current RTT value are regarded as input states. While in [70], the arrival time interval of ACK and minimum RTT are chosen as features and the AdaBoost algorithm is used to classify the loss into wireless loss and congestion loss for satellite networks. In [18], the one-way delay and inter-packet times were used as states to predict loss categories. In [19], the queuing delay, the inter-arrival time, and lists of packets were used as inputs. Besides, diverse supervised learning techniques were applied. In [71], decision trees, decision tree ensembles, bagging, random forests, extra-trees, boosting, and multi-layer perceptrons were used to classify the types of loss. Simulations show that these intelligent loss classifiers achieve high accuracy in different network scenarios. In conclusion, when classifying the wireless loss and the congestion loss, delay information is the kernel state.

Beyond wireless loss, contention loss is common in Optical Burst Switching (OBS) networks. OBS provides an advanced network, which saves the sources due to wavelength reservation. However, because of the lack of buffers in OBS, contention loss is generated when there is a burst at the core nodes. There are some supervised learning-based CC algorithms designed to tackle this. In [72], some classic contention resolutions are discussed and measured including wavelength conversion, deflection routing selection, and buffering with shared feedback fiber delay line. To measure the efficiency of these strategies, burst loss probability, and burst probability were considered. These strategies show good performances related to OBS contention issues. While in [73], a Hidden Markov Model was used to classify contention loss, congestion loss, and control congestion separately. Simulations showed the effectiveness of loss classifiers in different network scenarios.

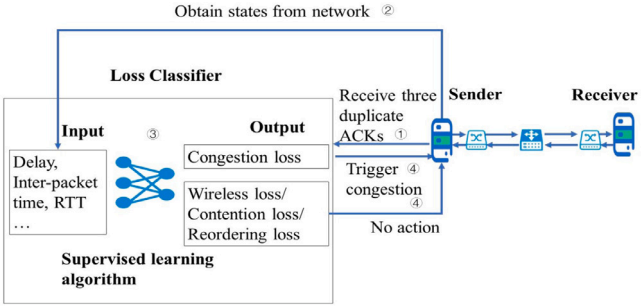

Fig. 1. Loss classification based on supervised learning algorithms.

Reordering loss cannot be ignored in networks with multi-channel paths. In networks, when packets are reordered, reordering loss occurs. Supervised learning-based CC algorithms are able to deal with the associated classification issues. In [74], out-of-order delivery causes variations of RTT. Therefore, RTT related to reordering and RTT related to congestion show different distributions. In [20], a Bayesian algorithm was used to represent the distributions of RTT for two types of losses. The proposed algorithm showed high prediction accuracy.

In conclusion, wireless loss, contention loss, and reordering loss impact the detection of congestion loss. Supervised learning techniques show advantages in classifying types of losses in different network scenarios. The mechanism is shown in Fig. 1 and Table 6 summarizes the studies related to loss classifiers based on supervised learning methods. However, there are some issues related to these supervised learning-based CC algorithms.

Misclassification is one issue. In wireless networks, predefined parameters determine the errors in classifying congestion loss and wireless loss. If the congestion loss is more easily classified than wireless loss, the classifier shows bad performances in wireless networks since the network is supposed to react when a loss is detected. However, due to the misclassification, the network considers congestion loss as wireless loss and does not control the sending rate quickly. Therefore, congestion cannot be reduced. Therefore, parameters in the algorithms need to be considered carefully to balance performance in different network scenarios.

The balance between computational complexity and prediction accuracy is another issue. As shown in [71], compared with decision trees, boosting algorithms achieve higher accuracy but consume much more network resources. Therefore, considering the limited improvements in the accuracy of boosting, the technique of decision trees shows more advantages due to its simplicity.

3.1.2. Delay prediction

As a congestion signal, the delay of transmissions reflects the amount of in-flight data, which shows the overall load on the network. There are some classic delay-based CC algorithms such as Vegas that measures delay accurately [4]. However, in dynamic networks, traditional delay-based CC algorithms are not flexible enough. As Fig. 2 shows and Table 7 concludes, supervised learning techniques have high

Table 6
Supervised Learning: Loss Classification in End-to-end CC Algorithms.

Algorithms	Scenarios	Input	Output
Decision tree boosting [18]	Wireless networks	One-way delay, inter-packet times	Link loss or congestion loss
Bayesian [20]	Networks with reordered events	RTT of lost packets	Reordering loss or congestion loss
Hidden Markov model [73]	OBS	Number of bursts successfully received at an egress between any two bursts	Contention loss or congestion loss
DT, Bagging, Boosting, Neural networks [19]	Wireless networks	Queuing delay, inter-arrival times, lists of packets	Wireless loss or Congestion loss
Multi-layer Perceptron [69]	Wireless networks	Minimum RTT, current RTT	Wireless loss or Congestion loss
AdaBoost [70]	Satellite networks	Minimum RTT, the arrival time interval of ACK	Wireless loss or Congestion loss
Decision trees, Decision tree Ensembles, Bagging, Random forests, Extra-trees, Boosting, Multilayer perceptrons, K-Nearest neighbors [71]	Wireless networks	Standard deviation, minimum, and maximum of one-way delay, inter-packet time for the packets	Wireless loss or Congestion loss

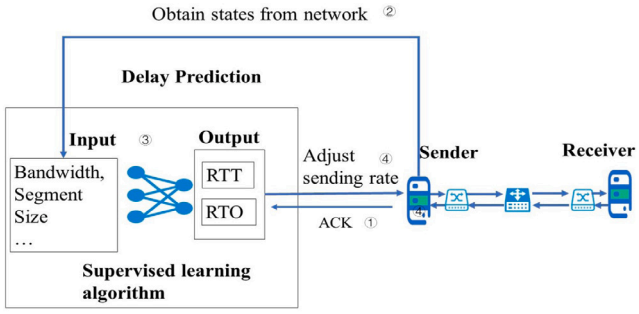

Fig. 2. Delay prediction based on supervised learning algorithms.

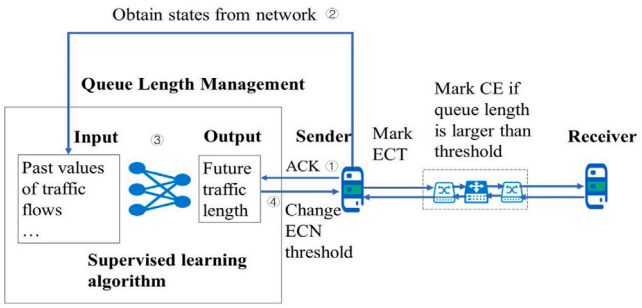

Fig. 3. Queue length management based on supervised learning algorithms.

learning capabilities and are efficient in predicting future delays and reacting quickly to avoid congestion.

RTT prediction is a major topic in delay prediction. Based on the measured RTT, other parameters can be calculated such as RTO. There has been a body of research exploring the prediction of RTO based on RTT. In [75], the estimation of RTT was dynamically changed to estimate RTO in wireless networks. In [76], RTT was used to predict RTO and bandwidth utilization. In [77], a fixed-share expert was used to compute the RTO in mobile and wired scenarios relying on RTT estimations. In addition, in [78] and [79], the fixed-share leveraged exponentially weighted moving average technique demonstrates a more accurate algorithm.

Moreover, there has been various research measuring RTT based on other parameters in the network. In [80], linear regression was used to establish the relationship between RTT and the sending rate. In [81], a Bayesian technique was used to simulate the distribution between delay and the sending rate and then to predict delay based on the sending rate. This is needed in real-time video applications and wireless networks.

Delay prediction is also significant for delay-sensitive networks that require networks with increased responsiveness. Several intelligent algorithms for the prediction of RTT using limited parameters and simple techniques to guarantee the low computational complexity and high responsiveness have been proposed. Further research is needed to push the boundary and deal with more complex related parameters and techniques to improve delay predictions.

3.2. Queue length management in network-assisted networks

Queue length management is a key focus for network-assisted CC algorithms. There has been a body of research related to the Active queue management (AQM) family of ECN techniques. However, the original AQM algorithms detect the current queue length and react to the environment. Some research has shown that the future queue length can be predicted. The prediction process is shown in Fig. 3. Moreover, Table 8 summarizes some related research. [88] and [89] showed the long-range dependence between previous traffic patterns and future queuing behavior. Multiple supervised learning techniques have been applied including Linear Minimum Mean Square Error Estimation [85], Normalized Least Mean Square Algorithm [86], Neural Networks [82, 83], Deep Belief Networks [87], and Neural-fuzzy [84].

These algorithms share similar features in that they employ the time series of previous traffic as input without considering diverse parameters in the network. As a result, these algorithms leave space for further exploration of dependencies between related parameters and the queue length.

4. Unsupervised learning-based congestion control algorithms

In this section, another category of learning-based CC algorithms is presented: unsupervised learning-based CC algorithms. Unsupervised learning techniques are used when the category of data is unknown, and the sample set needs to be clustered according to the similarity between samples in an attempt to minimize the intra-class gap and maximize the inter-class gap. In some situations, the network information cannot be fully provided, the unsupervised learning techniques can be alternatives, where the training data is not labeled. Thus, unsupervised learning methods can be employed as traffic classifiers as well. Thus, based on clusters generated by unsupervised learning methods, more network optimizations can be conducted such as CC. Clustering methods among unsupervised learning techniques include K-means [90], Hierarchical Clustering [91], Density-based Mean Shift [92], Density-based DBSCAN [93], and Expectation Maximization (EM) [94].

Table 7
Supervised learning: Delay measurement in End-to-end CC algorithms.

Algorithms	Scenarios	Details of the algorithms
Fixed-share experts [77]	Delay-sensitive networks	Employ the experts' framework to predict the RTT and then adjust the network environment to improve the goodput
Fixed-share with exponentially weighted moving average without increasing computational complexity [78]	Networks with fluctuating time scales	Propose a technique to estimate the RTT in scenarios with diversified RTT.
Bayesian theorem [81]	Real-time video applications and wireless networks	Adapt the sending rate based on the estimated delay
Linear regression [80]	Interactive video applications	Build a statistical function between the sending rate and RTT and adjust the sending rate based on the linear regression given the estimated RTT

Table 8
Supervised learning: Queue management in network-supported CC algorithms.

Algorithms	Scenarios	Details of the algorithms
Neural networks [82,83]	ATM networks	Predict the future value of the traffic based on the past traffic flows
Neural-fuzzy [84]	ATM networks	Use the estimated average queue length to calculate loss and then control the sending rate
Linear minimum mean square error estimation [85]	Networks supporting AQM	Establish a relationship between long-range traffic flows to estimate the future traffic based on past traffic flows
Normalized least mean square [86]	Networks supporting AQM	Employ adaptive techniques to estimate the instantaneous queue length
Deep belief networks [87]	NDN	Calculate the average queue length based on the prediction of pending interest table entries

Table 9
Unsupervised Learning: Loss Clustering in End-to-end CC algorithms.

Algorithms	Scenarios	Details of the algorithms
Hidden Markov Models [21], [95]	Wired/wireless networks	Uses delay-loss pairs to cluster data into several groups and assign the specific sending rate for each group
EM Clustering [73]	Optical burst switching networks	Cluster loss into contention loss and congestion loss and adjust the environment separately

K-means is easy to implement but it requires the user to specify the number of clusters in advance. Moreover, the clustering result is sensitive to the selection of the initial cluster center. In [90], K-means is used as a basic clustering method. The features selected in this work include the number of total packets, the number of actual data bytes, the number of the pushed data packets, etc. Based on K-means, the authors adopt the feature selection to find an optimal feature set and log transformation to improve the accuracy. The experiments show that the proposed method can obtain up to 80% overall accuracy.

Compared with K-means, EM is generally used for Gaussian Mixture Model. In K-means, an unknown data point must belong to a single cluster, while a data point can be mapped into multiple clusters based on EM. In [94], the authors use an AutoClass approach which is an EM model to cluster traffic based on statistical flow properties. In this research, a feature selection technique is used for finding the optimal flow attributes. The experiments show that the average accuracy is 86.5%. However, the performance of clustering depends on the particular application. This research shows that EM has more advantages than K-means in complex data sets, but this advantage is limited.

To compare different clustering methods, in [93], the authors compare K-means, AutoClass, and DBSCAN models to classify traffic by exploiting the distinctive characteristics of applications in the network communication. The experiments show that both K-means and DBSCAN work well and have faster training speed than AutoClass. As for the accuracy, K-means and AutoClass achieve high accuracy than DBSCAN. However, DBSCAN can produce better clusters than K-means and AutoClass. As a result, it is shown that in the situation of this research, these three algorithms have different advantages.

As for the detection of congestion, unsupervised learning algorithms are used to cluster loss groups and delay characteristics. In the next parts, more details will be presented.

4.1. Congestion detection in end-to-end congestion control algorithms

4.1.1. Loss clustering

In networking, unsupervised learning techniques are used to cluster loss into several groups and allocate resources for each group to achieve CC as shown in Fig. 4. The sender will collect data on the network side, such as RTT, and use an unsupervised learning algorithm to cluster the lost packets. The model will cluster the packet loss into congested packet loss and non-congested packet loss. When the current result is congested packet loss, a congestion signal will be triggered, and the sender will consider that congestion has occurred, and take corresponding measures to control the sending rate. A detailed summary is shown in Table 9.

According to previous literature, the packet delay variations reflect the available bandwidth and loss types. Thus, loss-delay pairs can be used to cluster the loss in networks [97]. In [21], the observation is that wireless losses and congestion losses can be differentiated by delays around losses. The distributions of round-trip delays around different types of loss have different features that can be used to classify

Table 10
Unsupervised learning: Delay clustering in End-to-end CC algorithms.

Algorithms	Scenarios	Details of the algorithms
K-means [96]	Vehicular ad hoc networks	Cluster the data into groups based on message size, the validity of messages, the distance between vehicles and RSUs, the types of message, and the direction of message and assign a sending rate for each cluster

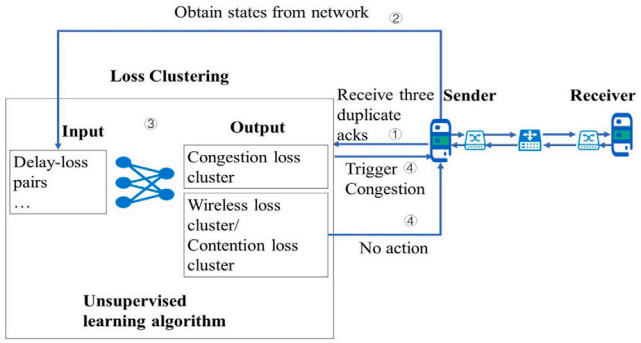

Fig. 4. Loss clustering based on unsupervised learning algorithms.

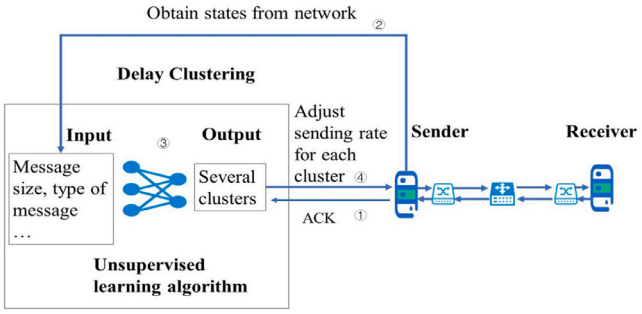

Fig. 5. Delay clustering based on unsupervised learning algorithms.

losses. The observation is that the delay distribution around wireless losses is different from the one around congestion losses. Based on the observation, in wireless networks, losses can be classified into congestion loss and wireless loss. In this research, the distribution of loss-delay pairs can be captured by a Hidden Markov Model (HMM). The HMM is trained to associate the type of losses with a state based on the delay features it captures. The experiments show that the proposed model is more efficient than Vegas predictors.

In addition to using the HMM to construct the conditional delay distribution caused by congestion or wireless channel errors, the Bayesian method is also added in [95] to train an effective classifier. In this research, the losses clustering problem is formulated as a statistical hypothesis testing. Besides, the Maximum Likelihood Ratio test is used as well. The analytical simulations show that as long as there are enough statistical data about the losses, a simple Bayesian method can be used to construct an effective classifier. If HMM is added, the clustering results will be more accurate, but the computational cost will be higher.

Except for the wireless losses mentioned above, unsupervised learning algorithms are used to cluster contention losses and congestion losses as well. In OBS, the authors in [73] find that the number of bursts between failures can be used to differentiate congestion and contention losses because the number of bursts' failures follows a Gaussian distribution with different parameters for contention and congestion losses. As for the models, HMM and EM are employed as clustering methods. Moreover, the authors modify the TCP with these two clustering methods to improve the performance of CC. In the experiments, results show that HMM-TCP and EM-TCP have higher throughput and goodput compared with traditional CCs including TCP NewReno, TCP SACK, and Burst TCP. Compared with HMM-TCP, EM-TCP performs slightly better among all metrics because the EM model has a higher degree of intra-cluster similarity.

As mentioned above, unsupervised learning algorithms are widely used for loss clustering. Though unsupervised learning techniques are simple and easy to implement, compared with loss classification based on supervised learning algorithms, the research related the loss clustering based on unsupervised learning algorithms is still relatively much less. For instance, the related techniques only cover EM and HMM. Network states mainly consist of loss-delay pairs and the number of bursts between failures. Thus, more research is required to broaden this field.

4.1.2. Delay prediction

There are only a limited number of unsupervised learning-based CC algorithms suitable for delay prediction because of the high processing demands for delay calculation. Typical algorithms such as k-means [96] and the associated mechanisms are presented in Fig. 5 and Table 10. As shown in Fig. 5, the sender obtains the states in networks such as the message size. Then, delay clustering is conducted based on unsupervised learning methods. The generated clusters can be used to adjust the sending rate based on the given policy. Specifically, Data such as the message size, the validity of messages, the distance between vehicles and RUSs, and the type of message is divided into different groups and the lowest delay in each group is selected as the communication parameter for each cluster. Based on the communication parameter, a specific sending rate will be assigned to each cluster. Therefore, based on the measurement of delay, CC can be achieved. Similar to loss clustering based on unsupervised learning techniques, the research of delay prediction relying on unsupervised learning methods is limited as well.

5. RL-based congestion control algorithms

RL algorithms typically include a value function and a policy function. The value function is responsible for measuring the value of specific actions given the network state, to determine if a given action can be chosen. The policy function is used to choose the action based on a given set of rules. In a given iteration, the system chooses an action based on the policy and the system provides feedback. The value function then calculates the value of the action and updates it accordingly. Based on different mechanisms, RL algorithms are divided into value-based schemes and policy-based schemes. Typical value-based schemes include Q Learning and DQL. Typical policy-based schemes include Policy-Gradient, Actor-Critic (AC), PPO, DDPG, and Asynchronous Advantage Actor-Critic (A3C). The difference between value-based schemes and policy-based schemes is that policy-based schemes estimate the policy for actions and whether they can satisfy scenarios with different actions, while value-based schemes predict the value of actions directly.

In the network communication, RL is widely used in specific scenarios such as 5G and edge computing to optimize network functions. In [105], a DRL approach is applied to deal with the issue of resource allocation in Millimeter-wave (mmWave) which is a key technology of fifth-generation wireless systems to achieve a high data rate. Due to the large bandwidth of mmWave and the high cost of infrastructure

Table 11
RL: Window Updating in End-to-End CC Algorithms (States, Actions, and Rewards)

Algorithms	Scenarios	States	Actions	Rewards
AC [98]	ATM networks	Taped delay-value of the number of the cells, and taped delay values of the feedback control signal	The coding rate	The input multiplexer buffer overflow, and the level of the coding rate of the input source
Q learning and Sarsa [99]	SDN	Link's occupied bandwidth	The sending rate	The threshold of congestion, and the occupied bandwidth of the link
DQL [34]	NDN	The prefix of requesting content, the priority, CWND, the total number of Interest packets sent, the total number of Data packets received, the total number of packets retransmitted, the average size of packets, the average RTT of packets, the time of monitor interval, and the time of decision interval	CWND	Throughput, RTT, loss, and reordering level
DDPG [36]	MPTCP in satellites communications	CWND, RTT, the number of packets failing to receive the ACK packet, and the cumulative rate number of retransmissions	CWND	CWND, RTT, the number of packets failing to receive the ACK packet, and the cumulative rate number of retransmissions
Fuzzy Kanerva-based Q Learning [100]	IoT	A moving average of the inter-arrival time between newly received ACKs, a moving average of the inter-arrival time between packets sent by the sender, the ratio between current RTT and the best RTT found, and the slow start threshold	CWND	Throughput and delay
Q learning [101]	Disruption tolerant networks	The input rate, the output rate, and available buffer space	CWND	The congestion level
Finite action-set learning automata [102]	AWNs	The inter-arrival times of ACK and DUPACK packets	CWND	The current inter-arrival time and the mean which computed over the inter-arrival times of the ACK packets arrived in the previous RTT
Continuous action-set learning automata [103]	AWNs	The inter-arrival times of TCP itacks	CWND	The mean and standard deviation of inter-arrival times of the ACK packets
DQL [25]	Wireless networks	The CWND difference, RTT, the minimum RTT over RTT ratio, the difference between RTT and the minimum RTT, and the inter-arrival time of ACKs	CWND	RTT and goodput
DDPG [104]	Network with time-varying flows	Goodput, average RTT, the mean deviation of RTTs and CWND	CWND	Goodput
Q learning [27]	Dynamic networking	The average interval between sending two packets, the average interval between receiving two consecutive ACKs, and the average RTT	CWND	Throughput and RTT
A3C [23]	Network with diversified flow size	Throughput, RTT, and losses	CWND	Throughput and RTT
PPO [35]	The dynamic networking	The derivative of latency concerning time, the ratio of the current MI's mean latency to minimum observed mean latency of any MI in the connection's history, and the ratio of packets sent to packets acknowledged by the receiver	The sending rate	Throughput, latency, and loss

upgrade, it is difficult to enhance the capacity of backhaul links between mmWave and the core network manually. Moreover, the data rates of mmWave users vary over time. Thus, to dynamically allocate the resources among users in backhaul links, the authors propose a DRL model to learn the dynamic environment. There is much research on resource management in other network environments based on DRL as well. For instance, in [106], DRL is regarded as the channel access strategy to improve the probabilities of successful transmission over multi-channel wireless networks. While in [107], DRL is used to predict channel qualities based on different base stations and then control handover in ultra-dense networks. Moreover, the resource management in edge computing and edge cache also uses the DRL model such as [108,109], and [110]. Therefore, RL, especially DRL, can play an important role in the network field including CC.

Amongst the different ML-based CC algorithms, RL has gained the most attention. Different from supervised learning methods, RL algorithms monitor the status of the environment continuously and react to the environment to optimize a utility function. Therefore, RL algorithms are more suitable for variable and unstable network environments. Two main trends are related to this kind of network. First, ubiquitous applications in data centers and cloud computing require efficient CC algorithms to deal with complicated network topologies [99]. In this context, reliability can be extremely important given the variances that can appear in the system. RL algorithms adapt to the errors promptly based on learning from the environment. Second, mobile devices such as smartphones, often connect to wireless networks including WIFI and 4G cellular in an ad hoc fashion. Thus, more flexible network topologies and diversified flows are a major challenge [27]. Traditional ML approaches are not dynamic enough to cope with diverse network environments based on trained models, unlike RL algorithms. These two trends are driving RL-based CC algorithms. In RL-based CC algorithms, the RL technique is used to update CWND based on different scenarios in end-to-end networks and to manage the queue length in network-assisted environments.

5.1. Window updating in end-to-end networks

Compared to supervised learning and unsupervised learning techniques, RL algorithms are more responsive to environmental changes. Instead of predicting congestion loss and delay as with supervised and

Table 12
RL: Window updating in End-to-End CC algorithms (Pros and Cons).

Algorithms	Pros	Cons
AC [98]	Can be a preventive CC, and the statistical multiplexing gain is enhanced	Limited state space and performance metrics
Q learning and Sarsa [99]	Routing information is considered, and the convergence speed is faster	Not suitable for complex networks due to the basic Q learning mode and the rough design of the reward function
DQL [34]	Comprehensive state parameters and well-designed reward function, High link utilization with limited packet loss, packet reordering, and latency	Heavy computational load
DDPG [36]	Deal with the high-dimensional state space and continuous action spaces with high efficiency and feasibility	Inadequate comparative experiments
Fuzzy Kanerva-based Q learning [100]	Dramatically reduce the memory requirements of a learning-based protocol while maintaining the same throughput and delay by using a function approximation method	Lack real deployment test
Q learning [101]	The delivery ratio is higher and end-to-end latency is lower compared with some representatives of the current DTN CC algorithms	More performance is supposed to be considered such as fairness and the packet loss rate
Finite action-set learning automata [102]	Not need any explicit feedback such as congestion, link failure, and available bandwidth notifications	Only the inter-arrival time of packets are used to measure the network conditions, and more parameters such as current CWND are necessary to obtain more accurate network conditions
Continuous action-set learning automata [103]	Low computational overhead, and implement the algorithm in a real testbed	There are many given parameters in the model while sensitivity analysis is not provided
DQL [25]	Show great performance in complex and dynamic network environments	The computational overhead may be an issue while there is no prototype implementation
DDPG [104]	Show great capability in dynamic flows based on LSTM, and flexible and robust to highly-dynamic networks with time-varying flows, and friendly to regular TCP	The convergence speed is not measured. Because the proposed model agent controls all active MPTCP flows, it is difficult to train a central agent
Q learning [27]	Use a small subset of sufficient information to accurately approximate the whole state space based on a novel generalization-based Kanerva coding approach	Lack real deployment test
A3C [23]	Fast convergence speed, high throughput, and low response time	The reward function is coarse-grained
PPO [35]	Show outstanding performance and robustness	Fairness, safety, and generalization are still challenges

unsupervised learning-based CC algorithms, RL-based CC algorithms learn the CC rules directly based on different environment information. Since RL algorithms can incorporate real-time network conditions and define actions accordingly, real-time control is possible in RL algorithms.

Various explorations have focused on RL-based CC algorithms that use RL to update CWND for specific scenarios. The mechanism of RL-based CC algorithms is shown in Fig. 6. the summary is shown in Tables 11 and 12, which shows more detailed information including states, rewards, actions, pros, cons, etc.

5.1.1. Asynchronous transfer mode networks

Asynchronous transfer mode (ATM) is a typical network suitable for RL-based CC algorithms. ATM networks are classic networks that support multi-media applications. For different multimedia traffic, ATM offers different QoS such as cell loss rate (CLR) and delay. However, in ATM, highly time-varying traffic patterns can increase the uncertainty of network traffic. Moreover, the small cell transmission time and low buffer sizes in ATM networks require more adaptive and high responsive CC algorithms. In [98], an AC algorithm is applied to deal with these problems. In the proposed CC algorithm, AC focuses on the performance function based on the CLR and voice quality. In each step, the algorithm measures the action according to the performance. In this way, different traffic patterns are connected with corresponding actions. Simulation results show that the CLR is low and voice quality is maintained. Compared with classical optimal control algorithms which rely upon a very accurate mathematical model in ATM, the proposed algorithm can understand the dynamics of network conditions and thus can minimize the CLR and maximize the level of the coding rate simultaneously. Moreover, since the algorithm is used at the input access node of the network and thus the speed of the algorithm is not limited by the propagation delay, any control action will avoid the potential congestion timely. Furthermore, because more sources can be supported for each multiplexer, the statistical multiplexing gain is enhanced. However, the limitation of this algorithm is that the proposed algorithm is relatively simplified. Both the state space and the feedback only consider the parameters to be optimized, without considering other variables such as traffic characteristics, so it may not be suitable for more complex environments.

5.1.2. Software defined networks

Software Defined Networks (SDNs) provide a new architecture for future networks that separate the forwarding and control planes. The control plane has the ability to manage the overall network centrally. Efficient CC algorithms are essential for SDNs. In [99], Q learning is used to tackle such advanced networks. In this research, a modified Q learning algorithm is employed and the improvement is embodied in two aspects. On the one hand, the optimized Q learning algorithm considers the routes of the current flow which affect the congestion of links. On the other hand, the authors add the congestion judgment. If the bandwidth occupancy of the link has reached the threshold and if the congestion has occurred, the training of the Q learning algorithm can

be stopped. Thus, the modified Q learning algorithm converges faster. The experimental results show that higher link utilization and lower congestion level can be achieved. While the limitation of the algorithm lies in the simple topology and coarse-grained reward functions. In the experiments, the topology includes a core layer switch and five edge layer. Because Q learning algorithms may be suitable for scenarios with limited state space due to the limited storage capacity of Q tables, the feasibility of the algorithm in complex networks cannot be verified. As for the reward function, only congestion level and the bandwidth occupancy are taken into consideration. More parameters such as one-way delay may also be used as the evaluation parameter. Thus, further exploration is supposed to be implemented.

5.1.3. Named data networking

Named Data Networking (NDN) is an emerging future network architecture as well. The main characteristic of NDN is connectionless, providing content perceptibility and in-network caching. Typical applications of NDN are mobile and real-time communications. Therefore, CC algorithms are expected to cope with diverse and dynamic content. In [34], the deep RL algorithm considers the diversity of different content and adds a prefix when requesting content into the network. Therefore, the variety of content is considered when a given action is taken. In this algorithm, the bandwidth utilization, delay, loss rate, and packet ordering are fully considered in the states. Moreover, in the reward function, the throughput, losses, RTT, and the reordering level are measured as well. Thus, the proposed algorithm can reach the bottleneck bandwidth quickly in the initial stage with a small overshoot compared with BBR and Iterative Closest Point algorithm. Besides, high link utilization during the stable phase with low latency, packet loss rate, and packet reordering can be achieved. Of course, due to the comprehensive parameters covered, the computational load is heavy.

5.1.4. Satellite communication networks

Satellite communication networks are dynamic and have time-varying flows. High bandwidth and high elasticity are key features. Video streaming is one representative application. In satellite communication networks, frequent satellite handover can be a severe problem, which may result in routing failures, packet blocking, and channel quality impacts. To deal with these problems, [36] employs DDPG to design a multi-path TCP. By measuring the retransmission rate of each sub-flow, the RTT and the number of packets failing to receive the ACK packet are considered and the algorithm degrades the possibility of handover. In addition, the DDPG model has the capability to deal with the high-dimensional state space and continuous action spaces in the satellite communication networks with high efficiency and feasibility. However, in this research, the experiments are not convincing because the algorithm is not compared with other MPTCP algorithms. Thus, some performances cannot be measured such as fairness.

5.1.5. Internet of things

The Internet of Things (IoT) is a product of rapidly evolving wireless technology. Some core features of IoT are local computation, the high variability of use, and potential computational demands. In [100], Q learning was used to satisfy diverse IoT networks with reduced computational needs with strong learning capabilities. The proposed algorithm was suitable for real-time processors and memory demands of IoT environments. In the designed model, traditional Kanerva coding is used to reduce the memory required to store the state–action value table. Based on the Kanerva coding, states and actions, are represented as state–action pairs. By choosing the prototype state–action pairs, memory requirements are dramatically reduced. In the simulations, the algorithm shows the capability to achieve high throughput, low delay, and good fairness. But the simulations are conducted in the NS3 simulator. While in the IoT environment, a real deployment test is essential to verify the performance of the learning-based algorithm in smart devices. Thereby, the feasibility in the real IoT environment is expected to be tested.

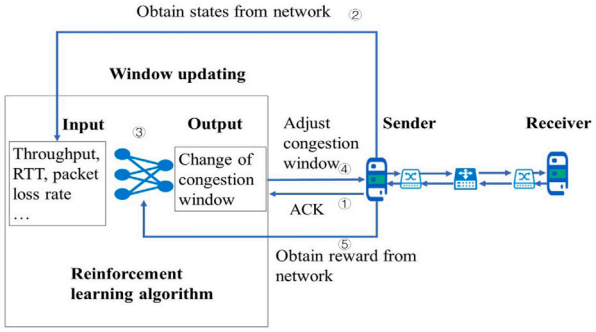

Fig. 6. Window updating based on RL algorithms.

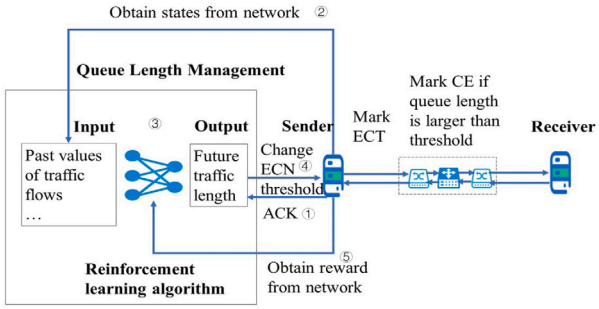

Fig. 7. Queue length management based on RL algorithms.

5.1.6. Disruption tolerant networks

Different from traditional networks, such as TCP Internet, Disruption Tolerant Networks (DTN) are subject to high latency due to very long propagation delays such as interplanetary communication. Therefore, the DTN cannot guarantee end-to-end connectivity between nodes and extremely long latencies. The network control in DTN is done on a hop-by-hop basis. Due to the high latency, the DTN nodes are required to store data in persistent storage for arbitrarily long periods before they find a suitable next-hop. Thereby, CC is significant to ensure the data transmissions with limited congestion in DTN. While existing DTN CC schemes do not show great performance due to their inability to adjust to dynamic networks. In [101], Q learning is employed to make CC decisions based on local knowledge such as buffer occupancy. As a result, compared with some representatives of the current DTN CC algorithms such as Storage Routing [116], the delivery ratio is higher, and end-to-end latency is lower. The limitation of the research is that more performance parameters are expected to be measured such as fairness and the packet loss rate.

5.1.7. Wireless networks

The wireless network is a typical scenario in learning-based CC algorithms. In [25], DQL is used to deal with the CC issue in wireless networks. There are three aspects of optimizations: feature selection, reward function, and a modified training process. To acquire the stable information as features, in this paper, the authors utilize a deep Convolutional Neural Network (CNN) concatenated with a Long Short Term Memory (LSTM) network as an automatically tuned filter to extract stable information. While in the reward function, accurate RTT measurement is important, which affects the convergence of the training process. In this research, the authors adopt a low-pass filter with a fixed window size for RTT measurements to remove frequent and small latency jitters. As for the training process, on the one hand, the model uses a relatively small experience replay buffer to mitigate the non-stationary nature of the local experience. On the other hand, the model has concurrent experience replay trajectories for sampling

Table 13
RL: Queue management in network-supported CC algorithms (States, Actions, and Rewards).

Algorithms	Scenarios	States	Actions	Rewards
PID controller [111]	Networks supporting AQM	The queue length, and the target buffer occupancy	The packet dropping/marking probability	Queue length error
Adaptive neuron PID [112]	Networks supporting AQM	The queue length, and the target buffer occupancy	The packet dropping/marking probability	Queue length error
Neural network PID controller [113]	Networks supporting AQM	The queue length, and the target buffer occupancy	The packet dropping/marking probability	Queue length error
Neuron RL [114]	Networks supporting AQM	The queue length error, and the rate error	The packet dropping/marking probability	Queue length error
Q Learning [115]	Networks supporting AQM	Current queue length, and packet drop probability	BLUE parameters	Throughput, and instantaneous queuing delay

Table 14
RL: Queue management in network-supported CC algorithms (Pros and Cons).

Algorithms	Pros	Cons
PID controller [111]	Efficient and stable, and converge fast for long delay networks	Only queue length is measured without considering other performance metrics such as throughput
Adaptive neuron PID [112]	Converge to queue length target fast and maintain a smaller queue length jitter	Limited performance metrics
Neural network PID controller [113]	Have faster response speed and better robustness compared with the typical PID and PID based on neural networks	Not take into account multiple network dynamics
Neuron RL [114]	Improve the robustness and dynamic performance	Limited performance is considered
Q Learning [115]	The speed and accuracy are improved	The computational complexity is a challenge for routers with limited memory and computational resources

training data. Based on the simulations in the NS3 simulator, the proposed algorithm achieves superior performance over five benchmark algorithms such as Cubic and Vegas in complex and dynamic network environments. However, there cover many models like the deep CNN and DQL. Thus the computational overhead may be an issue while there is no prototype implementation to verify whether this problem exists.

Among wireless networks, there is much research about AWNs. AWNs are a collection of mobile wireless nodes without any fixed infrastructure. Therefore, AWNs have constrained resources, limited processing, and unpredictable mobility. They are also highly dynamic. In [102], Finite Action-set Learning Automat, a learning automation whose unique feature is learning the network state faster with reduced information and negligible computational requirements, contains a finite number of actions. The algorithm takes effect in learning the dynamic wireless environment with limited consumed resources. Different from traditional CC algorithms, the proposed algorithm does not need any explicit feedback such as congestion, link failure, and available bandwidth notifications. Instead, the presented algorithm observes the occurrence of events including the arrival of ACK and duplicate ACK packets, and updates the CWND. The limitation of the research is that only the inter-arrival time of packets is used to measure the network conditions. Therefore, more parameters such as the current CWND are necessary to obtain more accurate network conditions.

While in [103], Continuous Action-set Learning Automata was applied in AWNs. The discretization of Finite Action-set Learning Automata may not be proper in all situations, e.g. the discretization can be too coarse or too fine-grained. Therefore, Continuous Action-set Learning Automata was introduced to deal with an infinite number of actions. It maintains an action probability distribution. The advanced algorithm achieves great performance in real networks. Furthermore, the state information to be maintained and the computational requirements are significantly low. But the proposed model relies on many given parameters that affect the performance of the model. For example, as for the learning parameter of the step size, lower values of this parameter improve the accuracy of learning while higher values cause the proposed model to adapt to the changes in the network rapidly. Thus sensitivity analysis is necessary in order to get a robust model.

5.1.8. Networks with dynamic traffic

The RL-based CC algorithms above focus on single scenarios, however, there are some RL-based designed for more complex (multiple) network scenarios. For instance, [23,27,104] and [35] utilize the RL algorithm to deal with congestion problems in networks with time-varying flows. These algorithms have advantages in representing states based on specific models. For instance, in [104], the authors employ DDPG combined with LSTM to represent highly-dynamic networks with time-varying flows. Moreover, the model is proved to be robust and efficient in simulations. While in [27], a novel generalization-based Kanerva coding approach is applied to approximate the whole state space, which only uses limited storage resources. In [23] and [35], A3C and PPO can deal with continuous states and fast convergence speed as well due to the sophisticated model architecture.

From the above, it can be seen that RL-based CC algorithms can satisfy diverse network scenarios with high adaptability and strong flexibility. However, there are some limitations. For instance, convergence is very hard to guarantee for continuous tasks and complex algorithms. In addition, state abstraction is challenging. Current algorithms require significant storage to store states and actions and demand considerable memory resources. Moreover, their computational complexity is relatively high. As a result, though RL algorithms show strong learning capabilities, realistic applications require further exploration due to the engineering issues identified.

5.2. Queue length management in network-assisted networks

For the queue length management of RL-based CC algorithms, RL is used to manage the queue length based on the current state as shown in Fig. 7. Tables 13 and 14 show the fine-grained information of these algorithms. In queue management, Proportional Integral Derivative (PID) is the most commonly applied RL technique. In [111–113], PID is used to maintain the queue length given the target threshold by calculating the dropping probability. In [111] and [112], the authors use adaptive neurons to tune parameters in PID to control queue length.

Based on the designed model, the queue length can converge fast and can be maintained at a low level with stability. However, in this research, only the queue length is measured without considering other performance metrics such as throughput. While in [113] and [114], a new AQM algorithm is modified with RL techniques. In [113], the AQM algorithm is presented based on a Fuzzy controller. The designed model can automatically compute the learning rate according to the current network state. The simulation shows that the model is superior to the PID based on neural networks on the queue stability, convergence speed, and time delay. In [114], a neuron RL model is applied to improve the robustness of AQM.

Besides, the Q learning technique can be used to learn the optimal parameters in CC models. For instance, in [115], the Q learning technique is applied to adjust the parameters in the BLUE which is an active queue management algorithm. In the proposed model, the current queue length is considered as a part of states. Based on a Q learning technique, the modified BULE improves the speed and accuracy at the cost of high complexity.

Compared with window updating for end-to-end networks, the queue length management for network-assisted CC algorithms requires more computational resources because multiple nodes can be used to control congestion such as routers in network-assisted networks. Therefore, it may be a burden for the network to support RL-based CC algorithms given the larger state space and high computational complexity. Besides, current queue length management based on RL techniques only covers limited state parameters such as the past queue length and buffer size. However, more parameters are required to improve the performance of RL-based CC algorithms.

6. Simulation setup

In this section, we introduce the simulation setup for RL-based CC algorithms as representatives of learning-based CC approaches. We conduct experiments based on realistic network environments with challenges caused by large delay and their high complexity. We perform experiments based on the NS3 platform and explore the performances of RL-based CC algorithms and traditional CC algorithms. In the NS3 platform, the computational process related to the RL algorithms is separated from data transmission in the pipeline. As a result, the computational complexity of RL algorithms has no impact on network communications.

In the following sections, we compare algorithms, performance metrics, and network environments.

6.1. Compared algorithms

In the simulation, three RL algorithms are chosen: DQL, DDPG, and PPO, as typical examples of RL algorithms. Generally, DQL is the simplest among these three algorithms, hence it is suitable for relatively simple environments. DDPG and PPO have stronger learning capabilities, and hence they can be applied in more complex scenarios. Considering the limited complexity of our network environment, these three algorithms are expected to perform similarly. To compare them with a benchmark algorithm, NewReno, Cubic, and BBR are selected. On the one hand, these three traditional CC algorithms are widely used and robust in modern networks. On the other hand, they have different advantages in varied network scenarios. For NewReno, it is conservative and stable. However, it is not suitable for dynamic networks. While Cubic is more aggressive but sensitive to random loss. As for BBR, it satisfies flexible network environments but harms throughput when computing with loss-based CC algorithms. Therefore, these algorithms are highly representative.

6.1.1. DQL-based congestion control algorithms

Different from Q Learning or State–action–reward–state–action (Sarsa) which considers the state as a discrete finite set, DQL can deal with large-scale problems. In the DQL algorithm, the value function is expressed by neural networks such as CNN, Recurrent Neural Networks (RNN), and LSTM. DQL is relatively simple compared with other Deep RL and can deal with relatively simple networks. Except for the convergence issue, DQL is promising because the model is lighter.

6.1.2. DDPG-based congestion control algorithms

DDPG is an optimized version of the AC algorithm, which combines policy-based methods and value-based methods. The actor part is used to approximate the policy function and is responsible for generating actions that interact with the environment. AC takes advantage of both mainstream RL algorithms, but they can be difficult to converge since the two neural networks in AC algorithms are related to each other and both need to update the gradient.

DDPG is another category of RL algorithm to deal with the convergence issue of AC. It employs experience reply and double networks. In satellite communications, a DDPG-based algorithm was designed to deal with multi-path CC problems and achieved a high degree of effectiveness [36].

Compared with DQL, DDPG has a stronger capability to train models in more complex environments. However, it is unsuitable in random environments. Also, training DDPG models can be more difficult.

6.1.3. PPO-based congestion control algorithms

PPO is a deep RL algorithm based on AC schemes. PPO is used to solve problems where the traditional policy gradient method is not good enough to determine the learning rate or step size. If the step size is too large, the policy will keep moving and will not converge. However, if the step size is too small, it is time-consuming. To deal with this problem, PPO limits the updating range of new policies by using the ratio between the new and old policy, making the policy gradient less sensitive to slightly larger step sizes. To achieve this, PPO uses an adaptive penalty to control the policy change. In this way, PPO provides an optimized AC algorithm as well as improving the efficiency of convergence. To adapt to the variable network conditions, such as changeable link flows and end-to-end latency, PPO is presented as an RL-guided CC algorithm [35]. Simulations show that the proposed algorithm outperforms traditional CC algorithms in different contexts by generating optimal policies.

PPO has proven to be an outstanding deep RL method and the combination with CC shows the potential of PPO in a wide array of network applications. However, there exist some challenges such as the speed of training a policy related to the parameter structures. As a result, the training efficiency of PPO can be a major issue.

6.1.4. NewReno

NewReno is a loss-based CC algorithm based on Reno. It offers a slow start, congestion avoidance, retransmission, and fast recovery. In our experiments, NewReno is used as one of the representatives of traditional CC algorithms, which is the default CC algorithm in NS3 platform as well.

6.1.5. Cubic

Cubic is an improved version based on Bic. By replacing the concave and convex window growth part of Bic which uses a binary search method to determine the growth scale of the CWND with a cubic function including concave and convex parts, the window adjustment algorithm of Bic is greatly simplified. This function retains the advantages of Bic, such as stability and scalability. At the same time, the window control is simplified and its TCP friendliness is enhanced. However, the limitation is that Cubic is very susceptible to random packet losses. In many current versions of the Linux kernel such as Linux kernel version 2.6.19, Cubic is a commonly used CC algorithm, so it is also used as one of the representatives of classic CC algorithms in this research.

6.1.6. BBR

Based on the detection of network bandwidth and delay, BBR has complementary learning capabilities in classic CC algorithms. BBR builds a detection model to obtain the instantaneous link available maximum bandwidth and minimum RTT. Compared with NewReno and Cubic, BBR achieves higher throughput with shallow buffers and random losses and substantially lower queuing delays with deep buffers. But BBR does not perform well when competing with other CC algorithms such as Cubic because Cubic tends to fill up buffers [39]. BBR has been implemented in many versions of Linux kernels such as Linux kernel version 4.9. In this research, BBR represents the algorithm with strong adaptive capabilities among classic CC algorithms.

6.2. Performance metrics

Based on the literature, the network cares about several critical parameters including throughput, RTT, packet loss rate, and fairness. Therefore, in our experiments, our performance metrics focus on these four parameters. Throughput counts the amount of data successfully transmitted in a given unit of time, measured in Mbps. RTT measures the data transfer time from the sender to the receiver based on the average RTT in seconds. Packet loss rate calculates the ratio of packet loss in a given time interval. Fairness measures the capability of coexisting with other algorithms.

6.3. Network environment

6.3.1. Internet

All simulations employ the same network topology, comprising the same dumbbell topology with the same access delay and bandwidth. We adopt the dumbbell topology for two reasons. On the one hand, the dumbbell topology is widely used in many mainstream CC algorithms including RL-based CC algorithms such as TCP-Deep ReInforcement learNing-based Congestion control (Drinc) [25] and RL-TCP [117], because it is simple and representative. On the other hand, a simple dumbbell topology can also simulate a complex network environment by dynamically adjusting the link's random packet loss, link bandwidth, and RTT. Moreover, this environment is more controllable, which is convenient for evaluating the effect of the algorithm. Thus, to simulate different network environments, the bottleneck bandwidth and bottleneck delays are varied in our experiments.

Based on previous research, learning-based CC algorithms are more suitable for high-speed networks such as satellite communications networks [53], ATM networks [82], and networks with time-varying flows [104]. We speculate that learning-based CC algorithms are suitable in networks with high BDP since they are more aggressive in making use of higher BDP. The BDP can be a critical parameter to measure the network as it is used to control congestion in BBR as well [39]. Therefore, we design three scenarios as shown in Table 15 to compare the performance of traditional CC algorithms and RL-based CC algorithms.

In the scenarios, there are two senders and two receivers in the dumbbell network. The access bandwidth is 1000 Mbps and the access delay is 0.01 ms. In our experiments, high BDP and low BDP are relative and not absolute. In Scenario I, the BDP is high and the bottleneck bandwidth is high. However, the bottleneck delay is low. In Scenario II, the BDP is high, but the bottleneck bandwidth is low and the bottleneck delay is high. In Scenario III, the BDP is low, but the bottleneck bandwidth and bottleneck delay are low.

In Scenario I, the bottleneck delay is set to 2.5 ms. The bottleneck bandwidth changes from 100M to 140M in 5 s. More specifically, the bottleneck bandwidth is 100M initially and incremented by 10M each second up to a maximum of 140M.

In Scenario II, the bottleneck delay is set to 25 ms. The bottleneck bandwidth changes from 10M to 50M in 15 s. Every three seconds, the bottleneck bandwidth increases by 10M. Because the bottleneck delay is longer, more simulation time is required in Scenario II compared to the scenario I. This allows observing the performance of different CC algorithms.

In Scenario III, the bottleneck delay is set to 2.5 ms, and the bottleneck bandwidth changes from 10M to 50M in 5 s, i.e. every second the bottleneck bandwidth increases by 10M.

6.3.2. States

States often vary in different research approaches. In DQL-based CC algorithms, states mainly focus on CWND differences, RTT, and the inter-arrival time of ACKs [25]. In the A3C framework, states are based on throughput, loss, and RTT [23]. In self-learning CC algorithms relying on DDPG, states are based on CWND, RTT, ACK, and the cumulative rate number of retransmissions of the sub-flow [118]. While in PPO, the states are designed in three parts: the latency gradient, the latency ratio, and the sending ratio [35]. No guaranteed rules are underpinning RL-based CC algorithms. According to previous literature, states are used to tackle two key areas: congestion signals including RTT, loss, ACK, throughput, and the parameter used to control congestion such as the CWND size and the sending rate. In CC algorithms, the environment adjusts the sending rate or the CWND size based on the congestion signals.

Considering the focus on performance metrics, the states considered here are throughput, RTT, packet loss rate, and fairness.

6.3.3. Actions

In the simulations, all adjustments are window-based. By adjusting the CWND size, there are different rules that are applied. In [117], there are four actions: −1, 0, +1, +3. When the action is −1, the CWND will decrease one packet size. In [27], three actions are designed: −1, 0, +10. The increasing action is more aggressive (up to 10). In [96], the action space is much larger. Seven actions are predefined: +1, ∗1.25, ∗1.5, 0, −1, ∗0.75, ∗0.5. When the action is ∗1.25, the size of the new CWND is 1.25 times the original CWND. In our experiments, we considered four actions: −1, 0, +1, +3, as aligned with [117].

6.3.4. Rewards

Similar to states, rewards can have different definitions as well. In a DQL-based CC algorithm, the utility function of reward is defined as shown below [96].

$$Utility(t) = \alpha_i * log(throughput_i(t)) - \beta_i * RTT_i(t) - \gamma_i * loss_i(t) - \delta * reordering_i(t) \quad (1)$$

In the PPO-based CC algorithm [35], the utility function is defined as shown below:

$$Utility = 10 * throughput - 1000 * latency - 2000 * loss \quad (2)$$

In a A3C-based algorithm [23], the utility function is given as: $log(throughput/RTT)$. In a DDPG-based algorithm, the utility function is more complicated [118] and given as:

$$Utility = \sum_i (\alpha CWND_t - \beta rtt_t - \epsilon rta_t - kack_t) \quad (3)$$

To define the reward, the objective of the simulation should be defined first. The reward is used for feedback of the action given the current state. Using this, it measures the performance of the action. Thus the reward is a reflection of the performance of actions. From the above, the definition of reward covers throughput, delay, and packet loss rate. Considering these factors, the reward includes RTT and throughput. The utility function is shown below where the value of the utility reward is based on [96]. The bandwidth in the equation means the bottleneck bandwidth. *MinRTT* means the minimum RTT of the pipeline. P is used for the packet loss rate.

$$Utility = log(throughput/(bandwidth)) - log(RTT - MinRTT) + log(1 - p) \quad (4)$$

Table 15
Simulation scenarios.

Scenarios	Experiment Setting	BDP
Scenario I	Access bandwidth: 1000M Access delay: 0.01 ms Bottleneck bandwidth: changing from 100M to 140M in 5 s (bottleneck bandwidth increases by 10M every second) Bottleneck delay: 2.5 ms	High
Scenario II	Access bandwidth: 1000M Access delay: 0.01 ms Bottleneck bandwidth: changing from 10M to 50M in 15 s (bottleneck bandwidth increases by 10M every three seconds) Bottleneck delay: 25 ms	High
Scenario III	Access bandwidth: 1000M Access delay: 0.01 ms Bottleneck bandwidth: changing from 10M to 50M in 5 s (bottleneck bandwidth increases by 10M every second) Bottleneck delay: 2.5 ms	Low

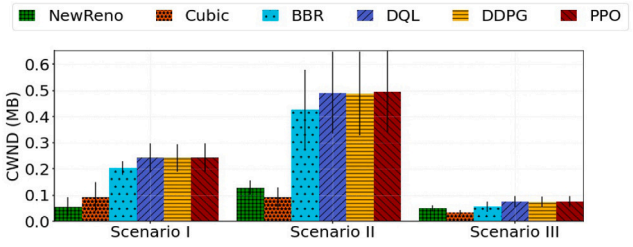

Fig. 8. CWND in the three scenarios.

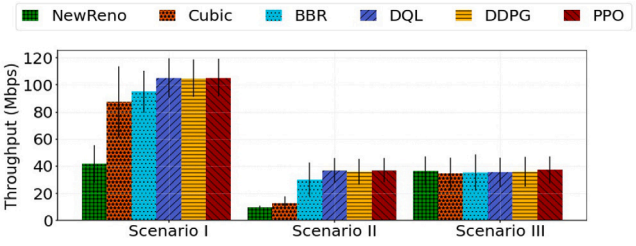

Fig. 9. Throughput in the three scenarios.

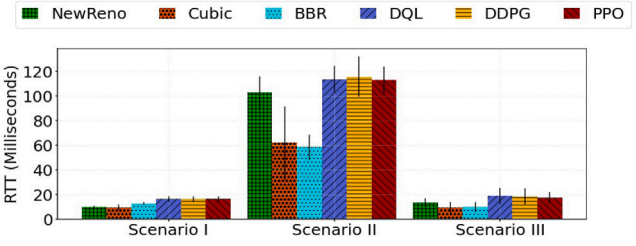

Fig. 10. RTT in the three scenarios.

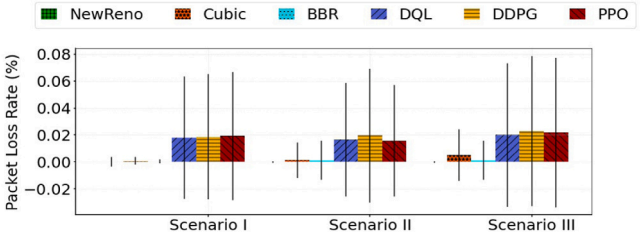

Fig. 11. Packet lost rate in the three scenarios.

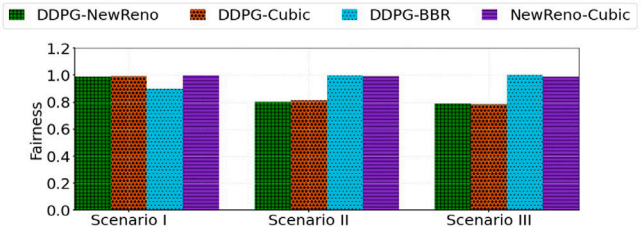

Fig. 12. Fairness in the three scenarios.

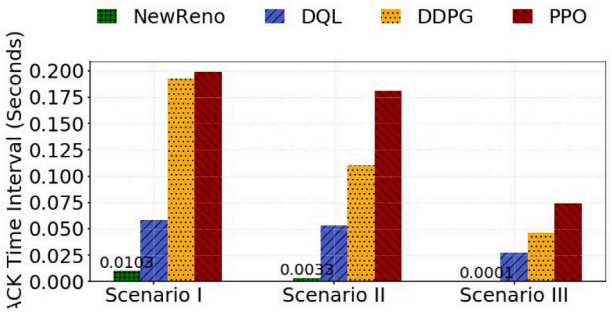

Fig. 13. ACK interval in realistic network simulation.

7. Simulations

In this section, we present the results of the simulations of the four algorithms: traditional CC algorithm NewReno and the RL-based CC algorithms, DQL, DDPG, and PPO. The simulations were conducted on the NS3 platform.

Based on previous research, the state space considered includes three parameters: throughput, RTT, and packet loss rate. The reward function is given as $log(throughput/bandwidth) - log(RTT - MinRTT) + log(1 - p)$. The action is used to adjust the CWND once a new ACK arrives. A dumbbell network topology was adopted.

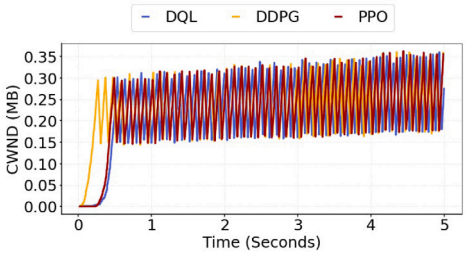

Fig. 14. Timeline of CWND in Scenario I.

Table 16
Simulation results.

Scenarios	BDP	CWND	Throughput	RTT	Packet loss rate
Scenario I	High	Substantial increase	Substantial Increase	Limited Increase	Limited increase
Scenario II	High	Substantial increase	Substantial Increase	Limited increase	Limited increase
Scenario III	Low	No big difference	No big difference	No big difference	Limited increase

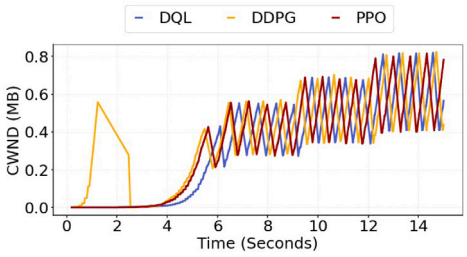

Fig. 15. Timeline of CWND in Scenario II.

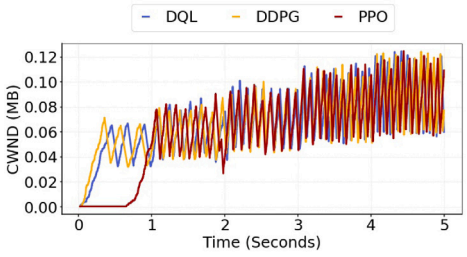

Fig. 16. Timeline of CWND in Scenario III.

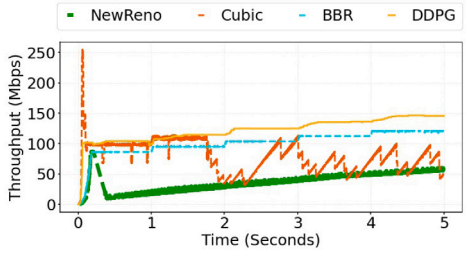

Fig. 17. Timeline of throughput in Scenario I.

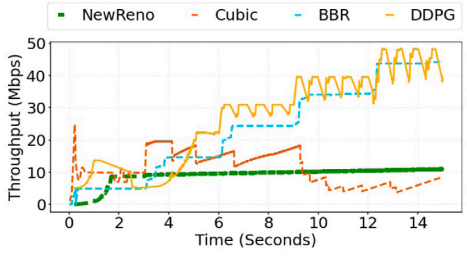

Fig. 18. Timeline of throughput in Scenario II.

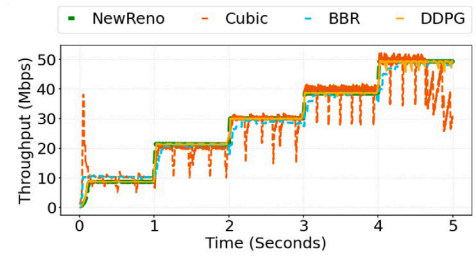

Fig. 19. Timeline of throughput in Scenario III.

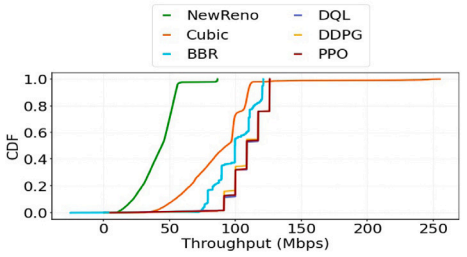

Fig. 20. CDF of throughput in Scenario I.

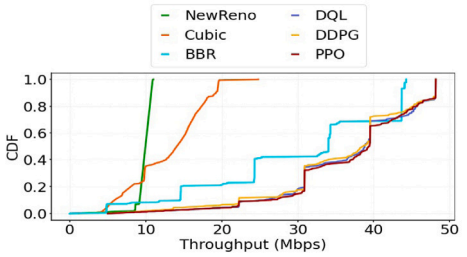

Fig. 21. CDF of throughput in Scenario II.

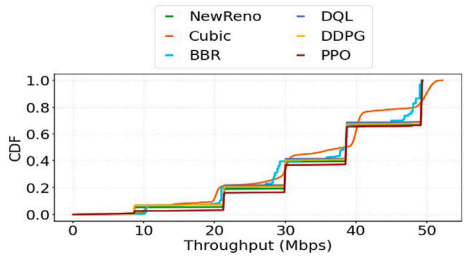

Fig. 22. CDF of throughput in Scenario III.

7.1. Simulation results

The overall simulation results are shown in Table 16 and Figs. 8 to 43 include timeline figures showing the changes of performances, bar figures showing the average and the variances of performances, and cumulative distribution function (CDF) figures showing the rough distributions of performances.

To check the performance of RL-based CC algorithms in realistic networks, we build an actual dumbbell topology with four Intel Xeon Gold servers and two commodity switches. Then, we use the Iperf [119] to generate actual flows based on the traces we applied in the simulation experiments. In the actual networks, the link bandwidth is 1Gbps and there is no change of the bandwidth. The base RTT is 0.3 ms. Our simulation time is 10 s. In order to verify the feasibility of RL-based CC algorithms, we compare them with NewReno. The result shows that the ACK interval is influenced by the computational complexity of the algorithms. As shown in Fig. 13, the ACK interval of RL-based CC algorithms is much larger than NewReno, resulting in a low growth rate of CWND. Since RL-based CC algorithms require considerable time to calculate and obtain the action, the ACK will not be transferred timely. It is noted that even amongst RL-based CC algorithms, there

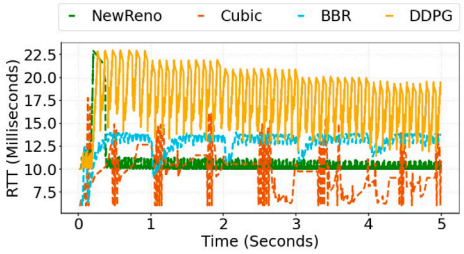

Fig. 23. Timeline of RTT in Scenario I.

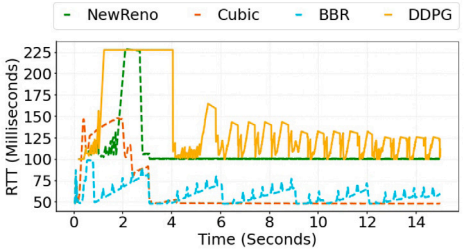

Fig. 24. Timeline of RTT in Scenario II.

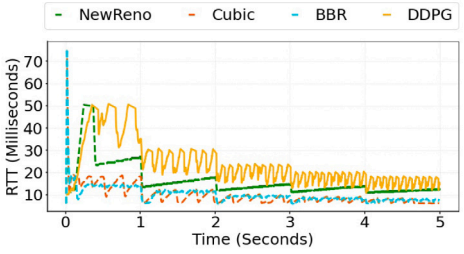

Fig. 25. Timeline of RTT in Scenario III.

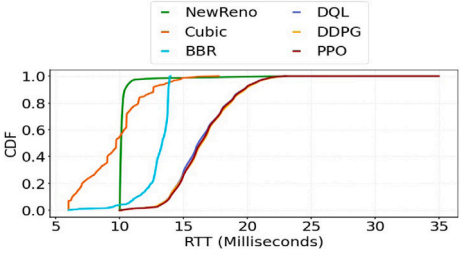

Fig. 26. CDF of RTT in Scenario I.

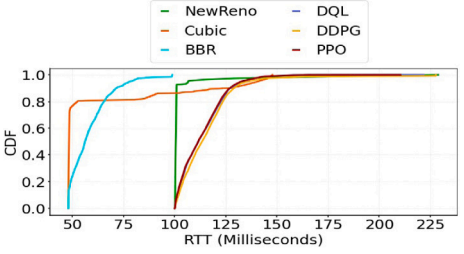

Fig. 27. CDF of RTT in Scenario II.

exist differences as well. Moreover, the delayed ACK influences the measurement of real throughput and RTT. Therefore, RL-based CC algorithms may be not applicable for realistic networks.

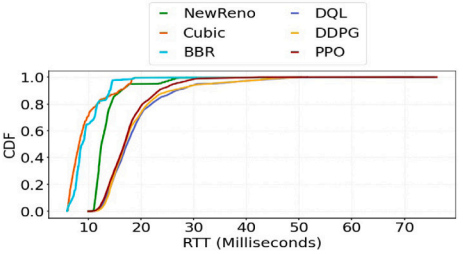

Fig. 28. CDF of RTT in Scenario III.

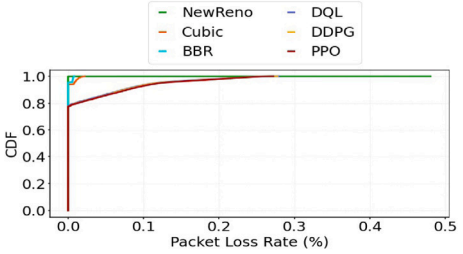

Fig. 29. CDF of loss rate in Scenario I.

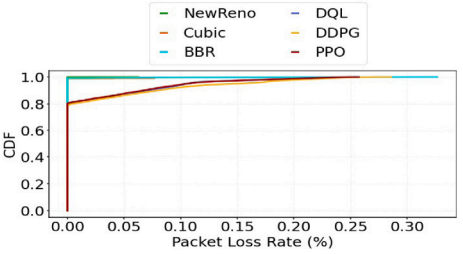

Fig. 30. CDF of loss rate in Scenario II.

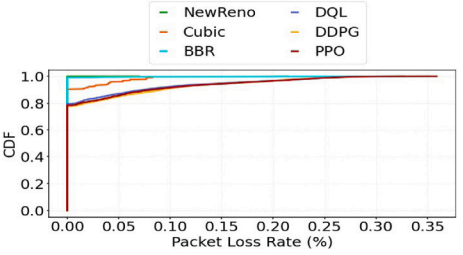

Fig. 31. CDF of loss rate in Scenario III.

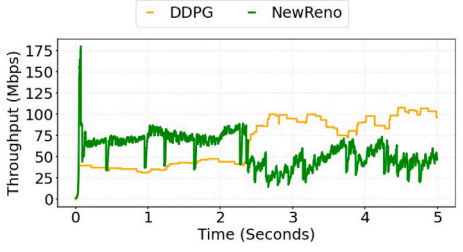

Fig. 32. Timeline of fairness of NewReno and DDPG in Scenario I.

To compare performances of RL-based CC algorithms and NewReno, we show the output based on NS3 where the delay caused by the RL algorithms is excluded. In the following sections, the detailed performance will be discussed including CWND and performance metrics.

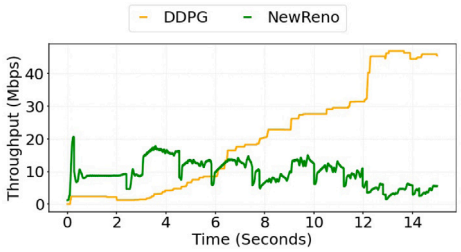

Fig. 33. Timeline of fairness of NewReno and DDPG in Scenario II.

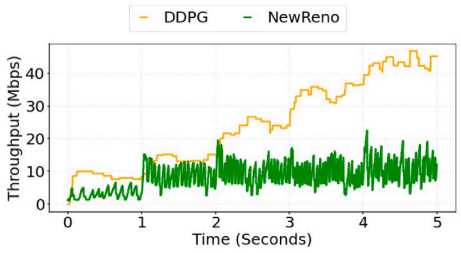

Fig. 34. Timeline of fairness of NewReno and DDPG in Scenario III.

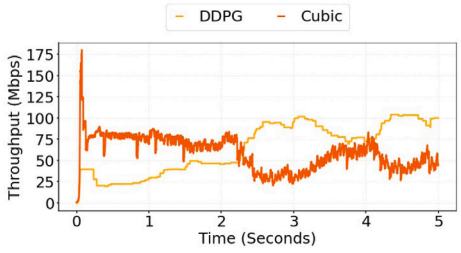

Fig. 35. Timeline of fairness of cubic and DDPG in Scenario I.

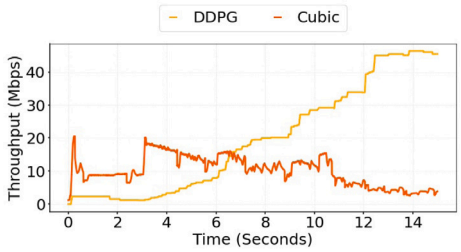

Fig. 36. Timeline of fairness of cubic and DDPG in Scenario II.

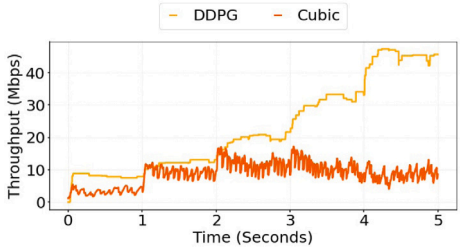

Fig. 37. Timeline of fairness of cubic and DDPG in Scenario III.

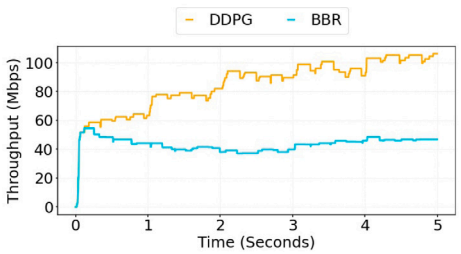

Fig. 38. Timeline of fairness of BBR and DDPG in Scenario I.

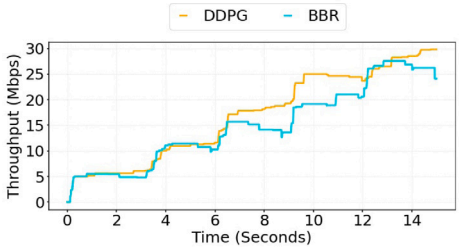

Fig. 39. Timeline of fairness of BBR and DDPG in Scenario II.

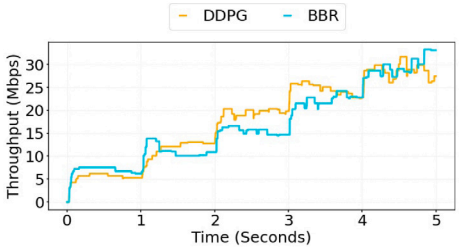

Fig. 40. Timeline of fairness of BBR and DDPG in Scenario III.

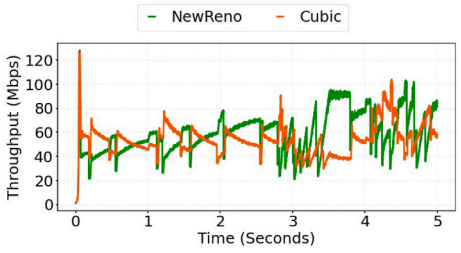

Fig. 41. Timeline of fairness of NewReno and Cubic in Scenario I.

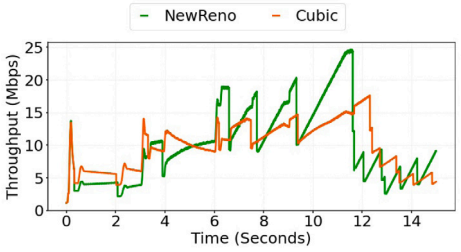

Fig. 42. Timeline of fairness of NewReno and Cubic in Scenario II.

7.1.1. CWND

Among the three RL-based CC algorithms, there is minimal difference between them in the three scenarios as shown in Fig. 8. Moreover, we observe that the size of CWND of RL-based CC algorithms is much larger than rule-based CC algorithms in Scenario I and the scenario II, which both have high BPD as expected. While in Scenario III, there is not much difference between these six algorithms. Figs. 14 to 16 have shown that the CWND of these three algorithms has similar changes

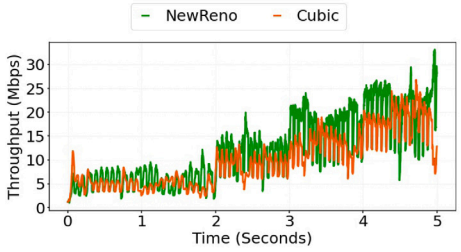

Fig. 43. Timeline of fairness of NewReno and Cubic in Scenario III.

in these three scenarios. Therefore, when we show the performance of RL-based CC algorithms later, we use DDPG among the three algorithms as a representative in some figures.

7.1.2. Throughput

Theoretically, the throughput of RL-based CC algorithms is expected to exceed the throughput of traditional CC algorithms such as NewReno, Cubic, and BBR due to the strong learning capability of RL-based CC algorithms in dynamic networks. As shown in Fig. 9, our speculation is verified. In Scenario I and the scenario II, throughput is improved when the RL-based CC algorithms are used. While in Scenario III, RL-based CC algorithms show no advantage. For the detailed distribution and timeline of throughput, more figures from Figs. 17 to 22 augment the results and explanation.

7.1.3. RTT

The RTT of NewReno is small and stable, which represents the benchmark of RTT. While the RTT of Cubic and BBR has a small fluctuation. In three scenarios, compared with NewReno, RTT is higher in networks with RL-based CC algorithms as shown in Fig. 10. From Figs. 23 to 28, among classic CC algorithms, because the growth rate of CWND among RL-based CC algorithms is more aggressive when the link available bandwidth increases suddenly, it is understandable that RTT is higher. While the RTT of these three classic algorithms is kept at a very low level. However, from the Figs. 23 to 25, it shows that increments of RTT are limited and bounded compared with increments of throughput in the three scenarios.

7.1.4. Packet loss rate

As shown in Fig. 11, the packet loss rate of classic CC algorithms is almost zero while there are minimal packet losses in networks with RL-based CC algorithms. Moreover, the distribution information shows the increased packet loss rate in RL-based CC algorithms from Figs. 29 to 31. Considering the aggressiveness of RL-based CC algorithms, bounded packet losses are understandable.

7.1.5. Fairness

In order to measure the fairness of RL-based CC algorithms, we employ a DDPG-based CC algorithm to compete with other classic CC algorithms. The benchmark experiment is the competition of NewReno and Cubic. As shown from Figs. 32 to 43, these competitions are conducted in three scenarios. The summary of fairness is shown in Fig. 12. Fig. 12 shows that the DDPG-based CC algorithm shows fantastic fairness compared with the benchmark experiment because the value of fairness exceeds 0.8. Moreover, BBR and the DDPG-based CC algorithm show excellent fairness when competing. Therefore, in terms of fairness, the DDPG-based CC algorithm performs relatively well.

7.2. Analysis of results

From the simulation results, it can be seen that RL-based CC algorithms can achieve high throughput with limited increased RTT and packet loss rate in networks with relatively high BDP. Moreover, in our network environments, three RL-based CC algorithms exhibited similar performance. Because the space complexity was not so high and the dynamic fluctuation was limited, these three algorithms handled these scenarios well. Therefore, our experiments showed that RL-based CC algorithms have advantages in high BDP networks (as simulated using NS3).

In realistic networks, CC algorithms react to the ACK arrival time. When a new ACK comes, the algorithm detects the delay or loss in the network and then adjusts the CWNDs or the sending rate. For traditional CC algorithms, there is a minor cost in time to compute the action because the adjustment rule is pre-designed and stable, while RL-based CC algorithms require lots of time to input the states to the neural network; get the output; update the action value and then take the appropriate actions. This process is time-consuming especially with the potential size of ACK transmission rates. As such, it is hard for RL-based CC algorithms to measure the actual transmission time of ACKs and almost impossible to measure the real network throughput, RTT, packet loss rate, and fairness. On the NS3 platform, these problems are not revealed because the NS3 platform separates the computation and transmission parts. Therefore, no matter how time-consuming the algorithm is, there is no impact on the ACK transmission. However, in real-world applications, such time must be considered. Thus whilst RL-based CC algorithms are applicable in the NS3 emulator, they are limited to realistic environments.

7.3. Proposed solutions

Based on the simulation results and analysis, it can be observed that current RL-based CC algorithms process rewards based on the arrival of ACKs, which are transferred and received one by one. As discussed, these RL-based CC algorithms are feasible on the NS3 simulator which separates the calculation and ACK transmission, however, the implementation of RL-based CC algorithms is still a problem. As a result, there are several possible future research trends.

Firstly, design lighter models based on mapping tables to deal with the problem of time-consuming RL-based CC algorithms. After an RL-based model is trained in network emulators, it can save the state and action to a table. Therefore, a mapping table can be prepared in advance. This process can be done off-line. When the model is deployed, only the mapping table is used. Given the state of the network environment, the action is given based on the mapping table. The time of this process is relatively small. This method can be efficient and time-saving. However, there are some challenges. The simple mapping table may be large and unwieldy in continuous scenarios. Therefore, more efficient mapping tables might be explored to address these limitations.

Another solution is decreasing the frequency of decisions, such as employing RL to select CC algorithms in a given time interval instead of selecting CWND size based on ACK arrival intervals. This means that the time interval for updating is much larger than the delay caused by the calculation of RL. Therefore, the impact of the delay caused by RL models can be ignored. Of course, the drawback is that the updating speed and responsiveness of the RL algorithm would be affected. Advanced research is proved to achieve high performance in dynamic environments with limited time and resource overhead [120]. Based on the traditional CC algorithm Cubic, the proposed algorithm computes the new CWND according to the DRL at regular intervals. Experiments in actual scenarios show that in a variety of network environments, such as intercontinental networks with high BDP, this algorithm has achieved suboptimal but relatively stable effects. Therefore, reducing the frequency of decisions of RL-based CC algorithms is a possible way to balance overhead issues and adaptability.

Finally, asynchronous RL algorithms are supposed to deal with delayed ACKs due to the algorithms' computational complexity. In an asynchronous RL framework, there are multiple actors. These actors take effect asynchronously, which can eliminate the effects of delayed ACKs. Therefore, in the network thread, ACKs are not blocked by the RL agent thread. In [121], to handle the delay of rewards, one action generates several partial actions. Therefore, each partial action can interact with the network environment independently. In addition, in [122], an asynchronous RL training framework, TorchBeast, combined with Pantheon network emulators, is used to handle delayed actions. The proposed algorithm, MVFST-RL, separates the network transmission and RL agents in realistic network communications based on multiple asynchronous actions. Though the algorithm eliminates the effect of delayed actions, the high resource-demanding training process is a problem since there are multiple actors to be trained and the state space is larger compared to synchronous RL-based CC algorithms. Therefore, the training process is more difficult. More research is required to address this issue.

8. Challenges and trends of learning-based congestion control schemes

8.1. Challenges of learning-based congestion control schemes

For traditional CC algorithms, the main issue is to detect congestion promptly and react quickly. The challenge of this kind of algorithm is the limited flexibility in dynamic networks. It is hard to satisfy different scenarios with a single algorithm. For learning-based CC algorithms, flexibility is improved but some issues need to be addressed.

Parameter selection influences the performance heavily, especially with RL algorithms. The state space [27,100], the action space [102, 103], the reward design, [34] and other hyper-parameters related to algorithm structures need to be considered carefully. Using reward design as an example. In an RL-based CC algorithm, throughput and RTT are used to calculate the reward [23]. While in other RL-based CC algorithms, throughput, packet loss rate, and delay are considered when calculating the reward [35]. For supervised learning, predefined parameters determine potential classification errors that affect the performance of CC [62]. For unsupervised learning algorithms, parameters such as the number of clustering groups and initial cluster centers influence the final clustering results [90]. Therefore, optimizing parameters is a non-trivial activity.

High computational complexity is a significant issue for learning-based CC algorithms. For supervised learning techniques, especially for hybrid and complex methods such as boosting and bagging, the prediction accuracy can be extremely high, but the computational complexity can also be high. For RL algorithms, the computational complexity results in delayed actions and rewards [122]. This impacts the utilization of bandwidth.

High memory consumption needs to be taken into consideration. The training of RL-based CC algorithms requires considerable storage space, especially for continuous network environments. Therefore, abstracting the state–action space and obtaining representative data is needed for an efficient training process. For example, LSTM [104] and Kanerva coding [27] are used to represent and abstract the network states. Some advanced RL frameworks such as DDPG [118] and A3C [121] have a strong capability to deal with continuous network environments by representing the state–action space using complex neural networks. Abstracting the representative state is thus the key. Currently, a huge space representation is a major limitation of complex scenarios.

Low training efficiency is related to the feasibility of deployment. For learning-based CC algorithms, the training process may be time-consuming and resource-consuming. State abstraction is important to improve training efficiency [27]. Optimal parameter selection can be helpful to improve the training efficiency as well. Tackling this issue requires more research. Current learning-based CC algorithms require significant amounts of training data to guarantee performance. However, though diverse network topologies and traffic flows can be simulated, the algorithms cannot always avoid over-fitting and under-fitting problems.

Hard convergence impacts RL-based CC algorithms. Considering complex algorithms with multiple neural networks, it can be difficult to attain convergence [104]. Current RL algorithms propose different approaches to contribute to convergence, however, for realistic networking, this cannot always be guaranteed.

Incompatibility is an open question requiring further research. Current learning-based CC algorithms are often used as a built-in component or an independent controller to control congestion [113]. There is still a long way to go for the issues related to compatibility between learning-based CC algorithms and traditional CC algorithms to be resolved.

Fairness cannot be guaranteed because the performance of ML-based CC algorithms relies on the trained model and the feedback from network environments. When the flow applying the ML-based CC algorithm competes with other flows, the fluctuations caused by other flows will be sensed by the ML-based CC algorithm. As a result, the ML-based algorithm may take inappropriate actions. For instance, as for the DDPG-based CC, in datacenter networks, competing with BBR and competing with Cubic show different fairness, which is shown in our simulation experiments. As for current RL-based CC algorithms, the flexibility of network environments is still limited [25,117]. Although these RL-based CC algorithms show good fairness in simulation experiments, fairness in actual complex scenarios cannot be guaranteed.

8.2. Trends of learning-based congestion control algorithms

Considering the issues associated with learning-based CC algorithms as mentioned above, several trends should be considered.

Firstly, engineering issues related to learning-based CC algorithms are a key research topic due to the high online capability of RL algorithms. Based on the previous literature, most learning-based CC algorithms are based on simulations using network emulators. On the one hand, simulations in network emulators can eliminate unrelated factors and are more suitable to design network scenarios. On the other hand, engineering issues can be ignored, e.g. parameter selection and computational complexity. In realistic network communications, such engineering issues are significant for learning-based CC algorithms. To design more applicable algorithms, simulations in realistic network environments will be a primary focus. However, in real networks, engineering issues are difficult to deal with because the latency in modern networks is smaller and smaller [12]. Thus, the model applied in real networks is expected to be highly responsive and resource-saving, which contradicts the high complexity of ML techniques.

In addition, advanced technologies such as programmable switches based on Programming Protocol-Independent Packet Processors (P4) language can be used to obtain network states to aid in the decision making in the end-host. For learning-based CC algorithms, the network state information is crucial. In traditional switches, the switch information is unavailable. Luckily, programmable switches are developed such as P4 switches [123]. P4 switches have widely applied to deal with network issues relying on the programmable capability. In [124], in the network with P4 switches, the header of the packet records the congestion information from the sender to the receiver, and there is only a small amount of bandwidth overhead. As a result, the detailed congestion information of the network can be obtained in advance. In [125], P4 switches are used to obtain network congestion status for rerouting. The P4 switches make the precise control in dynamic networks possible. In [126], the in-band network telemetry which is supported by P4 switches is applied to obtain network statuses such as the queue length and the timestamp at switches. Based on the

statuses that are used to compute the utilization ratio of links, end hosts can control the sending rate precisely. However, the combination of learning-based CC algorithms and P4 switches is still rare. Thus, there is much research space for learning-based CC algorithms combined with P4 switches.

Moreover, lightweight learning-based CC algorithms will be a hot topic in the future. Robust domain knowledge is needed to realize lightweight learning-based CC algorithms. Current learning-based CC algorithms have high complexity and can require considerable time to make decisions, with significant demands on memory and storage. Therefore, lighter-weight learning-based CC algorithms are required to be more applicable and deployable. To make models lighter, domain knowledge can be used to assist the designing of learning-based CC algorithms. Compared with the solid foundation of traditional CC algorithms which cover underlying theories such as RTT distributions in different scenarios and reordering schemes [74], current learning-based CC algorithms are relatively coarse-grained with limited knowledge support. They require a complete and detailed state space to train the model, making the model heavier. While to get lightweight models, utilizing fewer optimally states is necessary. The challenge of the designing of lightweight algorithms is that the ML model is more like a black box. It is not clear how domain knowledge affects the performance of the algorithm. Therefore, it is not easy to select effective parameters and design the models based on domain knowledge. Thus, simplifying the model is hard to achieve.

Finally, an open network platform that provides massively differentiated dynamic network scenarios supporting the exploration and evaluation of various learning-based CC algorithms is needed to facilitate further research in learning-based CC algorithms. Pantheon [127] belongs to this kind of platform. Though this platform covers diverse nodes, professional and specific network environments are not offered, e.g. flexible AWNs. Therefore, there is a demand for a general platform providing a professional and realistic simulation environment to train learning-based CC algorithms. In this way, the development of learning-based algorithms will be faster. This kind of platform requires amounts of computing resources including servers and switches, and at the same time designing stable and efficient interfaces to be suitable for various learning-based algorithms is necessary. Therefore, whether it is from a hardware perspective or a software perspective, the implementation of this kind of platform is a challenge.

9. Conclusion

Due to the limitations of traditional CC algorithms in dynamic networks, learning-based CC algorithms have seen a recent trend in academia. In this paper, we provided a review of state of the art in learning-based CC algorithms together with simulations focused on different RL-based CC algorithms as representatives of learning-based CC algorithms are conducted. In the simulations, it was shown that RL-based CCs algorithms exhibit better performances compared to traditional CC algorithms in different scenarios such as networks with high bandwidth and low delay. We presented and discussed limitations with current RL-based CC algorithms for realistic deployments and outline some approaches that may be used in future research. We identified challenges and trends associated with learning-based CC algorithms including dealing with engineering issues related to learning-based CC algorithms. In the future, network environments are expected to be increasingly complicated. Given this, there is a clear need for addressing such complexity and flexibility. To improve the performance and robustness, further research is required to deal with issues such as computation time, data storage, and pre-designed parameters. We argue that lightweight and efficient learning-based models with general learning-based platforms are needed and will be a future research focus.

CRediT authorship contribution statement

Huiling Jiang: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Qing Li:** Conceptualization, Supervision, Writing - review & editing. **Yong Jiang:** Funding acquisition. **GengBiao Shen:** Conceptualization, Supervision, Writing - review & editing. **Richard Sinnott:** Writing - review & editing. **Chen Tian:** Supervision. **Mingwei Xu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by Guangdong Province Key Area R&D Program under grant No. 2018B010113001, National Natural Science Foundation of China under grant No. 61972189, the project "PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)" and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

References

[1] J. Postel, Rfc0793: Transmission Control Protocol, RFC Editor, 1981.

[2] V. Jacobson, Congestion avoidance and control, in: SIGCOMM '88, Proceedings of the ACM Symposium on Communications Architectures and Protocols, Stanford, CA, USA, August 16-18, 1988, 1988, pp. 314–329, [Online]. Available: https://doi.org/10.1145/52324.52356.

[3] T.R. Henderson, S. Floyd, A.V. Gurtov, Y. Nishida, The newreno modification to tcp's fast recovery algorithm, RFC 6582 (2012) 1–16, [Online]. Available: https://doi.org/10.17487/RFC6582.

[4] L.S. Brakmo, S.W. O'Malley, L.L. Peterson, TCP vegas: New techniques for congestion detection and avoidance, in: Proceedings of the ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications, London, UK, August 31 - September 2, 1994, 1994, pp. 24–35, [Online]. Available: https://doi.org/10.1145/190314.190317.

[5] J. Sing, B. Soh, TCP New vegas: improving the performance of TCP vegas over high latency links, in: Fourth IEEE International Symposium on Network Computing and Applications, IEEE, 2005, pp. 73–82.

[6] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (4) (1993) 397–413, [Online]. Available: https://doi.org/10.1109/90.251892.

[7] S.S. Kunniyur, R. Srikant, Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management, in: Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, 2001, San Diego, CA, USA, 2001, pp. 123–134, [Online]. Available: https://doi.org/10.1145/383059.383069.

[8] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, Y. Zhu, L. Cui, Congestion control for cross-datacenter networks, in: 27th IEEE International Conference on Network Protocols, ICNP 2019, Chicago, IL, USA, October 8-10, 2019, 2019, pp. 1–12, [Online]. Available: https://doi.org/10.1109/ICNP.2019.8888042.

[9] S. Floyd, Highspeed TCP for large congestion windows, RFC 3649 (2003) 1–34, [Online]. Available: https://doi.org/10.17487/RFC3649.

[10] C. Caini, R. Firrincieli, TCP Hybla: a TCP enhancement for heterogeneous networks, Int. J. Satell. Commun. Netw. 22 (5) (2004) 547–566, [Online]. Available: https://doi.org/10.1002/sat.799.

[11] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control (BIC) for fast long-distance networks, in: Proceedings IEEE INFOCOM 2004, the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004, 2004, pp. 2514–2524, [Online]. Available: https://doi.org/10.1109/INFCOM.2004.1354672.

[12] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats, et al., TIMELY: RTT-based congestion control for the datacenter, in: ACM SIGCOMM Computer Communication Review, Vol. 45, ACM, 2015, pp. 537–550, no. 4.

[13] C.P. Fu, S.C. Liew, TCP veno: TCP enhancement for transmission over wireless access networks, IEEE J. Sel. Areas Commun. 21 (2) (2003) 216–228, [Online]. Available: https://doi.org/10.1109/JSAC.2002.807336.

[14] R. King, R.G. Baraniuk, R.H. Riedi, TCP-africa: an adaptive and fair rapid increase rule for scalable TCP, in: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA, 2005, pp. 1838–1848, [Online]. Available: https://doi.org/10.1109/INFCOM.2005.1498463.

[15] K. Ramakrishnan, S. Floyd, A proposal to add explicit congestion notification (ECN) to IP, RFC 2481 (1999) 1–25, [Online]. Available: https://doi.org/10.17487/RFC2481.

[16] D. Shan, F. Ren, Improving ECN marking scheme with micro-burst traffic in data center networks, in: 2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017, 2017, pp. 1–9, [Online]. Available: https://doi.org/10.1109/INFOCOM.2017.8057181.

[17] J. Zhang, W. Bai, K. Chen, Enabling ECN for datacenter networks with RTT variations, in: Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies, CoNEXT 2019, Orlando, FL, USA, December 09-12, 2019, 2019, pp. 233–245, [Online]. Available: https://doi.org/10.1145/3359989.3365426.

[18] I. El Khayat, P. Geurts, G. Leduc, Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes, in: International Conference on Research in Networking, Springer, 2005, pp. 549–560.

[19] I. El Khayat, P. Geurts, G. Leduc, Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning, Wirel. Netw. 16 (2) (2010) 273–290.

[20] N. Fonseca, M. Crovella, Bayesian packet loss detection for TCP, in: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA, 2005, pp. 1826–1837, [Online]. Available: https://doi.org/10.1109/INFCOM.2005.1498462.

[21] J. Liu, I. Matta, M. Crovella, End-to-end inference of loss nature in a hybrid wired/wireless environment, 2003.

[22] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C.H. Liu, D. Yang, Experience-driven networking: A deep reinforcement learning based approach, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 1871–1879.

[23] X. Nie, Y. Zhao, Z. Li, G. Chen, K. Sui, J. Zhang, Z. Ye, D. Pei, Dynamic TCP initial windows and congestion control schemes through reinforcement learning, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1231–1247, [Online]. Available: https://doi.org/10.1109/JSAC.2019.2904350.

[24] N. Jay, N.H. Rotman, B. Godfrey, M. Schapira, A. Tamar, A deep reinforcement learning perspective on internet congestion control, in: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, 2019, pp. 3050–3059, [Online]. Available: http://proceedings.mlr.press/v97/jay19a.html.

[25] K. Xiao, S. Mao, J.K. Tugnait, TCP-drinc: Smart congestion control based on deep reinforcement learning, IEEE Access 7 (2019) 11892–11904, [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2892046.

[26] S. Ryu, C. Rump, C. Qiao, Advances in internet congestion control, IEEE Commun. Surv. Tutor. 5 (1) (2003) 28–39, [Online]. Available: https://doi.org/10.1109/COMST.2003.5342228.

[27] W. Li, F. Zhou, K.R. Chowdhury, W. Meleis, QTCP: Adaptive congestion control with reinforcement learning, IEEE Trans. Netw. Sci. Eng. 6 (3) (2019) 445–458, [Online]. Available: https://doi.org/10.1109/TNSE.2018.2835758.

[28] N. Jay, N.H. Rotman, P.B. Godfrey, M. Schapira, A. Tamar, Internet congestion control via deep reinforcement learning, 2018, CoRR, vol. abs/1810.03259, [Online]. Available: http://arxiv.org/abs/1810.03259.

[29] M.A. Alsheikh, S. Lin, D. Niyato, H.-P. Tan, Machine learning in wireless sensor networks: Algorithms, strategies, and applications, IEEE Commun. Surv. Tutor. 16 (4) (2014) 1996–2018.

[30] M. Bkassiny, Y. Li, S.K. Jayaweera, A survey on machine-learning techniques in cognitive radios, IEEE Commun. Surv. Tutor. 15 (3) (2013) 1136–1159, [Online]. Available: https://doi.org/10.1109/SURV.2012.100412.00017.

[31] P.V. Klaine, M.A. Imran, O. Onireti, R.D. Souza, A survey of machine learning techniques applied to self-organizing cellular networks, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2392–2431.

[32] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O.M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, J. Internet Serv. Appl. 9 (1) (2018) 16.

[33] P. Gawlowicz, A. Zubow, Ns3-gym: Extending openai gym for networking research, 2018, CoRR, vol. abs/1810.03943, [Online]. Available: http://arxiv.org/abs/1810.03943.

[34] D. Lan, X. Tan, J. Lv, Y. Jin, J. Yang, A deep reinforcement learning based congestion control mechanism for NDN, in: 2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019, 2019, pp. 1–7, [Online]. Available: https://doi.org/10.1109/ICC.2019.8761737.

[35] N. Jay, N.H. Rotman, B. Godfrey, M. Schapira, A. Tamar, A deep reinforcement learning perspective on internet congestion control, in: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, 2019, pp. 3050–3059, [Online]. Available: http://proceedings.mlr.press/v97/jay19a.html.

[36] Z. Ji, Self-learning congestion control of MPTCP in satellites communications, in: IWCMC 2019, 2019.

[37] S. Floyd, T.R. Henderson, The newreno modification to tcp's fast recovery algorithm, RFC 2582 (1999) 1–12, [Online]. Available: https://doi.org/10.17487/RFC2582.

[38] S. Ha, I. Rhee, L. Xu, CUBIC: a new TCP-friendly high-speed TCP variant, Oper. Syst. Rev. 42 (5) (2008) 64–74, [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105.

[39] N. Cardwell, Y. Cheng, C.S. Gunn, S.H. Yeganeh, V. Jacobson, BBR: congestion-based congestion control, Commun. ACM 60 (2) (2017) 58–66, [Online]. Available: https://doi.org/10.1145/3009824.

[40] T. Benson, A. Akella, D.A. Maltz, Network traffic characteristics of data centers in the wild, in: M. Allman (Ed.), Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010, ACM, 2010, pp. 267–280, [Online]. Available: https://doi.org/10.1145/1879141.1879175.

[41] M. Allman, V. Paxson, W.R. Stevens, TCP congestion control, RFC 2581 (1999) 1–14, [Online]. Available: https://doi.org/10.17487/RFC2581.

[42] V. Jacobson, Congestion avoidance and control, Comput. Commun. Rev. 25 (1) (1995) 157–187, [Online]. Available: https://doi.org/10.1145/205447.205462.

[43] V. Jacobson, Modified TCP congestion avoidance algorithm, 1990, Email to the end2end-interest mailing list.

[44] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP selective acknowledgment options, RFC 2018 (1996) 1–12, [Online]. Available: https://doi.org/10.17487/RFC2018.

[45] S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, R. Wang, TCP westwood: Bandwidth estimation for enhanced transport over wireless links, in: MOBICOM 2001, Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, Rome, Italy, July 16-21, 2001, 2001, pp. 287–297, [Online]. Available: https://doi.org/10.1145/381677.381704.

[46] M. Hock, F. Neumeister, M. Zitterbart, R. Bless, TCP lola: Congestion control for low latencies and high throughput, in: 2017 IEEE 42nd Conference on Local Computer Networks (LCN), IEEE, 2017, pp. 215–218.

[47] C. Jin, D. Wei, S.H. Low, J. Bunn, H.D. Choe, J.C. Doylle, H. Newman, S. Ravot, S. Singh, F. Paganini, et al., FAST TCP: From theory to experiments, IEEE Netw. 19 (1) (2005) 4–11.

[48] S. Shalunov, G. Hazel, J.R. Iyengar, M. Kühlewind, Low extra delay background transport (LEDBAT), RFC 6817 (2012) 1–25, [Online]. Available: https://doi.org/10.17487/RFC6817.

[49] V. Arun, H. Balakrishnan, Copa: Practical delay-based congestion control for the internet, in: Proceedings of the Applied Networking Research Workshop, ANRW 2018, Montreal, QC, Canada, July 16-16, 2018, 2018, p. 19, [Online]. Available: https://doi.org/10.1145/3232755.3232783.

[50] K. Tan, J. Song, Q. Zhang, M. Sridharan, A compound TCP approach for high-speed and long distance networks, in: INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain, 2006, [Online]. Available: https://doi.org/10.1109/INFOCOM.2006.188.

[51] G. Marfia, C.E. Palazzi, G. Pau, M. Gerla, M.Y. Sanadidi, M. Roccetti, TCP *Libra* : Exploring RTT-fairness for TCP, in: I.F. Akyildiz, R. Sivakumar, E. Ekici, J.C. de Oliveira, J. McNair (Eds.), NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4479, Springer, 2007, pp. 1005–1013, [Online]. Available: https://doi.org/10.1007/978-3-540-72606-7_86.

[52] G. Carlucci, L.D. Cicco, S. Holmer, S. Mascolo, Analysis and design of the google congestion control for web real-time communication (webrtc), in: Proceedings of the 7th International Conference on Multimedia Systems, MMSys 2016, Klagenfurt, Austria, May 10-13, 2016, 2016, pp. 13:1–13:12, [Online]. Available: https://doi.org/10.1145/2910017.2910605.

[53] K. Winstein, H. Balakrishnan, TCP ex machina: computer-generated congestion control, in: ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013, 2013, pp. 123–134, [Online]. Available: https://doi.org/10.1145/2486001.2486020.

[54] C. Xu, B. Tu, G. Li, T. Yuan, J. Yang, Dual channel adaptive congestion control for datacenters, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019, IEEE, 2019, pp. 514–521, [Online]. Available: https://doi.org/10.1109/INFCOMW.2019.8845317.

[55] M. Dong, Q. Li, D. Zarchy, P.B. Godfrey, M. Schapira, PCC: Re-architecting congestion control for consistent high performance, in: 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015, 2015, pp. 395–408, [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/dong.

[56] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, M. Schapira, PCC vivace: Online-learning congestion control, in: 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018, 2018, pp. 343–356, [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/dong.

[57] T. Meng, N.R. Schiff, P.B. Godfrey, M. Schapira, PCC proteus: Scavenger transport and beyond, in: H. Schulzrinne, V. Misra (Eds.), SIGCOMM '20: Proceedings of the 2020 Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, Virtual Event, USA, August 10-14, 2020, ACM, 2020, pp. 615–631, [Online]. Available: https://doi.org/10.1145/3387514.3405891.

[58] K. Ramakrishnan, S. Floyd, D.L. Black, The addition of explicit congestion notification (ECN) to IP, RFC 3168 (2001) 1–63, [Online]. Available: https://doi.org/10.17487/RFC3168.

[59] P. Thaler, IEEE 802.1 Qau Congestion Notification, Citeseer, 2006.

[60] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, M. Yasuda, Less is more: Trading a little bandwidth for ultra-low latency in the data center, in: S.D. Gribble, D. Katabi (Eds.), Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012, USENIX Association, 2012, pp. 253–266, [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/alizadeh.

[61] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M.H. Yahia, M. Zhang, Congestion control for large-scale RDMA deployments, in: S. Uhlig, O. Maennel, B. Karp, J. Padhye (Eds.), Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015, ACM, 2015, pp. 523–536, [Online]. Available: https://doi.org/10.1145/2785956.2787484.

[62] M. Roughan, S. Sen, O. Spatscheck, N.G. Duffield, Class-of-service mapping for qos: a statistical signature-based approach to IP traffic classification, in: A. Lombardo, J.F. Kurose (Eds.), Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference, IMC 2004, Taormina, Sicily, Italy, October 25-27, 2004, ACM, 2004, pp. 135–148, [Online]. Available: https://doi.org/10.1145/1028788.1028805.

[63] A.W. Moore, D. Zuev, Internet traffic classification using bayesian analysis techniques, in: D.L. Eager, C.L. Williamson, S.C. Borst, J.C.S. Lui (Eds.), Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2005, June 6-10, 2005, Banff, Alberta, Canada, ACM, 2005, pp. 50–60, [Online]. Available: https://doi.org/10.1145/1064212.1064220.

[64] J. Park, H. Tyan, C.J. Kuo, Internet traffic classification for scalable QOS provision, in: Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, ICME 2006, July 9-12 2006, Toronto, Ontario, Canada, IEEE Computer Society, 2006, pp. 1221–1224, [Online]. Available: https://doi.org/10.1109/ICME.2006.262757.

[65] N. Jing, M. Yang, S. Cheng, Q. Dong, H. Xiong, An efficient SVM-based method for multi-class network traffic classification, in: S. Zhong, D. Dou, Y. Wang (Eds.), 30th IEEE International Performance Computing and Communications Conference, IPCCC 2011, Orlando, Florida, USA, November 17-19, 2011, IEEE Computer Society, 2011, pp. 1–8, [Online]. Available: https://doi.org/10.1109/PCCC.2011.6108074.

[66] S. Biaz, N.H. Vaidya, Discriminating congestion losses from wireless losses using inter-arrival times at the receiver, in: Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. ASSET'99 (Cat. No. PR00122), IEEE, 1999, pp. 10–17.

[67] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, H. Tokuda, Achieving moderate fairness for UDP flows by path-status classification, in: Proceedings 25th Annual IEEE Conference on Local Computer Networks. LCN 2000, IEEE, 2000, pp. 252–261.

[68] S. Cen, P.C. Cosman, G.M. Voelker, End-to-end differentiation of congestion and wireless losses, IEEE/ACM Trans. Netw. 11 (5) (2003) 703–717, [Online]. Available: https://doi.org/10.1109/TNET.2003.818187.

[69] K. Han, J.Y. Lee, B. Kim, Machine-learning based loss discrimination algorithm for wireless TCP congestion control, in: International Conference on Electronics, Information, and Communication, ICEIC 2019, Auckland, New Zealand, January 22-25, 2019, IEEE, 2019, pp. 1–2, [Online]. Available: https://doi.org/10.23919/ELINFOCOM.2019.8706382.

[70] N. Li, Z. Deng, Q. Zhu, Q. Du, Adaboost-TCP: A machine learning-based congestion control method for satellite networks, in: 19th IEEE International Conference on Communication Technology, ICCT 2019, Xi'an, China, October 16-19, 2019, IEEE, 2019, pp. 1126–1129, [Online]. Available: https://doi.org/10.1109/ICCT46805.2019.8947121.

[71] P. Geurts, I.E. Khayat, G. Leduc, A machine learning approach to improve congestion control over wireless computer networks, in: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK, 2004, pp. 383–386, [Online]. Available: https://doi.org/10.1109/ICDM.2004.10063.

[72] C.M. Gauger, M. Kohn, J. Scharf, Comparison of contention resolution strategies in OBS network scenarios, in: Proceedings of 2004 6th International Conference on Transparent Optical Networks (IEEE Cat. No. 04EX804), Vol. 1, IEEE, 2004, pp. 18–21.

[73] A. Jayaraj, T. Venkatesh, C.S.R. Murthy, Loss classification in optical burst switching networks using machine learning techniques: improving the performance of TCP, IEEE J. Sel. Areas Commun. 26 (6-Supplement) (2008) 45–54, [Online]. Available: https://doi.org/10.1109/JSACOCN.2008.033508.

[74] V. Paxson, End-to-end internet packet dynamics, IEEE/ACM Trans. Netw. 7 (3) (1999) 277–292, [Online]. Available: https://doi.org/10.1109/90.779192.

[75] W. Lou, C. Huang, Adaptive timer-based TCP control algorithm for wireless system, in: 2005 International Conference on Wireless Networks, Communications and Mobile Computing, Vol. 2, IEEE, 2005, pp. 935–939.

[76] P. Karn, C. Partridge, Improving round-trip time estimates in reliable transport protocols, Comput. Commun. Rev. 25 (1) (1995) 66–74, [Online]. Available: https://doi.org/10.1145/205447.205455.

[77] B.A.A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, K. Obraczka, A machine learning framework for TCP round-trip time estimation, EURASIP J. Wireless Commun. Networking 2014 (1) (2014) 47.

[78] Y. Edalat, J.S. Ahn, K. Obraczka, Smart experts for network state estimation, IEEE Trans. Netw. Serv. Manage. 13 (3) (2016) 622–635, [Online]. Available: https://doi.org/10.1109/TNSM.2016.2586506.

[79] Y. Edalat, J.S. Ahn, K. Obraczka, Network state estimation using smart experts, in: 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MOBIQUITOUS 2014, London, United Kingdom, December 2-5, 2014, 2014, pp. 11–19, [Online]. Available: https://doi.org/10.4108/icst.mobiquitous.2014.257949.

[80] T. Dai, X. Zhang, Y. Zhang, Z. Guo, Statistical learning based congestion control for real-time video communication, 2019, CoRR, vol. abs/1905.05998, [Online]. Available: http://arxiv.org/abs/1905.05998.

[81] T. Dai, X. Zhang, Z. Guo, Learning-based congestion control for internet video communication over wireless networks, in: IEEE International Symposium on Circuits and Systems, ISCAS 2018, 27-30 May 2018, Florence, Italy, 2018, pp. 1–5, [Online]. Available: https://doi.org/10.1109/ISCAS.2018.8351530.

[82] W. Jang, J. Byun, M.L. Hambaba, An intelligent architecture for ATM traffic congestion control, J. Intell. Fuzzy Systems 5 (2) (1997) 155–165, [Online]. Available: https://doi.org/10.3233/IFS-1997-5206.

[83] B. Hariri, N. Sadati, NN-RED: an AQM mechanism based on neural networks, Electron. Lett. 43 (19) (2007) 1053–1055.

[84] S.-J. Lee, C.-L. Hou, A neural-fuzzy system for congestion control in ATM networks, IEEE Trans. Syst. Man Cybern. B 30 (1) (2000) 2–9.

[85] Y. Gao, G. He, J.C. Hou, On exploiting traffic predictability in active queue management, in: Proceedings IEEE INFOCOM 2002, the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA, June 23-27, 2002, 2002, pp. 1630–1639, [Online]. Available: https://doi.org/10.1109/INFCOM.2002.1019416.

[86] A. Jain, A. Karandikar, R. Verma, An adaptive prediction based approach for congestion estimation in active queue management (APACE), in: Proceedings of the Global Telecommunications Conference, 2003. GLOBECOM '03, San Francisco, CA, USA, 1-5 December 2003, 2003, pp. 4153–4157, [Online]. Available: https://doi.org/10.1109/GLOCOM.2003.1259009.

[87] T. Liu, M. Zhang, J. Zhu, R. Zheng, R. Liu, Q. Wu, ACCP: adaptive congestion control protocol in named data networking based on deep learning, Neural Comput. Appl. 31 (9) (2019) 4675–4683, [Online]. Available: https://doi.org/10.1007/s00521-018-3408-2.

[88] A. Erramilli, O. Narayan, W. Willinger, Experimental queueing analysis with long-range dependent packet traffic, IEEE/ACM Trans. Netw. 4 (2) (1996) 209–223, [Online]. Available: https://doi.org/10.1109/90.491008.

[89] W. Willinger, V. Paxson, M.S. Taqqu, Self-similarity and heavy tails: Structural modeling of network traffic, Pract. Guide Heavy Tails Stat. Tech. Appl. 23 (1998) 27–53.

[90] Y. Liu, W. Li, Y. Li, Network traffic classification using K-means clustering, in: Proceeding of the Second International Multi-Symposium of Computer and Computational Sciences (IMSCCS 2007), August 13-15, 2007, the University of Iowa, Iowa City, Iowa, USA, IEEE Computer Society, 2007, pp. 360–365, [Online]. Available: https://doi.org/10.1109/IMSCCS.2007.52.

[91] C.S.M. Babou, D. Fall, S. Kashihara, Y. Taenaka, M.H. Bhuyan, I. Niang, Y. Kadobayashi, Hierarchical load balancing and clustering technique for home edge computing, IEEE Access 8 (2020) 127593–127607, [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3007944.

[92] D. Tang, J. Man, L. Tang, Y. Feng, Q. Yang, WEDMS: an advanced mean shift clustering algorithm for ldos attacks detection, Ad Hoc Netw. 102 (2020) 102145, [Online]. Available: https://doi.org/10.1016/j.adhoc.2020.102145.

[93] J. Erman, M.F. Arlitt, A. Mahanti, Traffic classification using clustering algorithms, in: Proceedings of the 2nd Annual ACM Workshop on Mining Network Data, MineNet 2006, Pisa, Italy, September 15, 2006, ACM, 2006, pp. 281–286, [Online]. Available: https://doi.org/10.1145/1162678.1162679.

[94] S. Zander, T.T.T. Nguyen, G.J. Armitage, Automated traffic classification and application identification using machine learning, in: 30th Annual IEEE Conference on Local Computer Networks (LCN 2005), 15-17 November 2005, Sydney, Australia, Proceedings, IEEE Computer Society, 2005, pp. 250–257, [Online]. Available: https://doi.org/10.1109/LCN.2005.35.

[95] D. Barman, I. Matta, Model-based loss inference by tcp over heterogeneous networks, in: Proceedings of WiOpt, 2004, pp. 364–73.
[96] N. Taherkhani, S. Pierre, Centralized and localized data congestion control strategy for vehicular ad hoc networks using a machine learning clustering algorithm, IEEE Trans. Intell. Transp. Syst. 17 (11) (2016) 3275–3285, [Online]. Available: https://doi.org/10.1109/TITS.2016.2546555.
[97] V. Paxson, Measurements and analysis of end-to-end internet dynamics, 1997.
[98] A.A. Tarraf, I.W. Habib, T.N. Saadawi, Reinforcement learning-based neural network congestion controller for ATM networks, in: Proceedings of MILCOM'95, Vol. 2, IEEE, 1995, pp. 668–672.
[99] R. Jin, J. Li, X. Tuo, W. Wang, X. Li, A congestion control method of SDN data center based on reinforcement learning, Int. J. Commun. Syst. 31 (17) (2018) [Online]. Available: https://doi.org/10.1002/dac.3802.
[100] W. Li, F. Zhou, W. Meleis, K.R. Chowdhury, Learning-based and data-driven TCP design for memory-constrained IoT, in: International Conference on Distributed Computing in Sensor Systems, DCOSS 2016, Washington, DC, USA, May 26-28, 2016, 2016, pp. 199–205, [Online]. Available: https://doi.org/10.1109/DCOSS.2016.8.
[101] A.P. Silva, K. Obraczka, S. Burleigh, C.M. Hirata, Smart congestion control for delay- and disruption tolerant networks, in: 13th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2016, London, United Kingdom, June 27-30, 2016, 2016, pp. 1–9, [Online]. Available: https://doi.org/10.1109/SAHCN.2016.7733018.
[102] V. Badarla, B.S. Manoj, C.S.R. Murthy, Learning-TCP: A novel learning automata based reliable transport protocol for ad hoc wireless networks, in: 2nd International Conference on Broadband Networks (BROADNETS 2005), 3-7 October 2005, Boston, Massachusetts, USA, 2005, pp. 521–530, [Online]. Available: https://doi.org/10.1109/ICBN.2005.1589652.
[103] V. Badarla, C.S.R. Murthy, Learning-TCP: A stochastic approach for efficient update in TCP congestion window in ad hoc wireless networks, J. Parallel Distrib. Comput. 71 (6) (2011) 863–878, [Online]. Available: https://doi.org/10.1016/j.jpdc.2010.12.012.
[104] Z. Xu, J. Tang, C. Yin, Y. Wang, G. Xue, Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1325–1336, [Online]. Available: https://doi.org/10.1109/JSAC.2019.2904358.
[105] M. Feng, S. Mao, Dealing with limited backhaul capacity in millimeter-wave systems: A deep reinforcement learning approach, IEEE Commun. Mag. 57 (3) (2019) 50–55, [Online]. Available: https://doi.org/10.1109/MCOM.2019.1800565.
[106] O. Naparstek, K. Cohen, Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–7.
[107] Z. Wang, Y. Xu, L. Li, H. Tian, S. Cui, Handover control in wireless systems via asynchronous multi-user deep reinforcement learning, 2018, CoRR, vol. abs/1801.02077, [Online]. Available: http://arxiv.org/abs/1801.02077.
[108] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, M. Bennis, Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning, IEEE Internet Things J. 6 (3) (2019) 4005–4018, [Online]. Available: https://doi.org/10.1109/JIOT.2018.2876279.
[109] Y. Sun, M. Peng, S. Mao, Deep reinforcement learning-based mode selection and resource management for green fog radio access networks, IEEE Internet Things J. 6 (2) (2019) 1960–1971, [Online]. Available: https://doi.org/10.1109/JIOT.2018.2871020.
[110] Z. Chang, L. Lei, Z. Zhou, S. Mao, T. Ristaniemi, Learn to cache: Machine learning for network edge caching in the big data era, IEEE Wirel. Commun. 25 (3) (2018) 28–35, [Online]. Available: https://doi.org/10.1109/MWC.2018.1700317.
[111] J. Sun, S. Chan, K.-T. Ko, G. Chen, M. Zukerman, Neuron PID: a robust AQM scheme, in: Proceedings of ATNAC, Vol. 2006, Citeseer, 2006, pp. 259–262.
[112] J. Sun, M. Zukerman, An adaptive neuron AQM for a stable internet, in: NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007, Proceedings, 2007, pp. 844–854, [Online]. Available: https://doi.org/10.1007/978-3-540-72606-7_72.
[113] Q. Yan, Q. Lei, A new active queue management algorithm based on self-adaptive fuzzy neural-network PID controller, in: 2011 International Conference on Internet Technology and Applications, IEEE, 2011, pp. 1–4.
[114] C. Zhou, D. Di, Q. Chen, J. Guo, An adaptive AQM algorithm based on neuron reinforcement learning, in: 2009 IEEE International Conference on Control and Automation, IEEE, 2009, pp. 1342–1346.
[115] S. Masoumzadeh, G. Taghizadeh, K. Meshgi, S. Shiry, Deep blue: A fuzzy q-learning enhanced active queue management scheme, in: 2009 International Conference on Adaptive and Intelligent Systems, IEEE, 2009, pp. 43–48.
[116] M. Seligman, K.R. Fall, P. Mundur, Alternative custodians for congestion control in delay tolerant networks, in: Proceedings of the 2006 SIGCOMM Workshop on Challenged Networks, CHANTS@SIGCOMM 2006, Pisa, Italy, September 11-15, 2006, ACM, 2006, pp. 229–236, [Online]. Available: https://doi.org/10.1145/1162654.1162660.
[117] Y. Kong, H. Zang, X. Ma, Improving TCP congestion control with machine intelligence, in: Proceedings of the 2018 Workshop on Network Meets AI & ML, NetAI@SIGCOMM 2018, Budapest, Hungary, August 24, 2018, 2018, pp. 60–66, [Online]. Available: https://doi.org/10.1145/3229543.3229550.
[118] T. Mai, H. Yao, Y. Jing, X. Xu, X. Wang, Z. Ji, Self-learning congestion control of MPTCP in satellites communications, in: 15th International Wireless Communications & Mobile Computing Conference, IWCMC 2019, Tangier, Morocco, June 24-28, 2019, 2019, pp. 775–780, [Online]. Available: https://doi.org/10.1109/IWCMC.2019.8766465.
[119] V.J.D. Barayuga, W.E.S. Yu, Packet level TCP performance of NAT44, NAT64 and IPv6 using iperf in the context of IPv6 migration, in: 5th International Conference on IT Convergence and Security, ICITCS 2015, Kuala Lumpur, Malaysia, August 24-27, 2015, IEEE Computer Society, 2015, pp. 1–3, [Online]. Available: https://doi.org/10.1109/ICITCS.2015.7293006.
[120] S. Abbasloo, C. Yen, H.J. Chao, Classic meets modern: a pragmatic learning-based congestion control for the internet, in: H. Schulzrinne, V. Misra (Eds.), SIGCOMM '20: Proceedings of the 2020 Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, Virtual Event, USA, August 10-14, 2020, ACM, 2020, pp. 632–647, [Online]. Available: https://doi.org/10.1145/3387514.3405892.
[121] M. Bachl, T. Zseby, J. Fabini, Rax: Deep reinforcement learning for congestion control, in: 2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019, 2019, pp. 1–6, [Online]. Available: https://doi.org/10.1109/ICC.2019.8761187.
[122] V. Sivakumar, T. Rocktäschel, A.H. Miller, H. Küttler, N. Nardelli, M. Rabbat, J. Pineau, S. Riedel, MVFST-RL: An asynchronous RL framework for congestion control with delayed actions, 2019, arXiv preprint arXiv:1910.04054.
[123] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, P4: programming protocol-independent packet processors, Comput. Commun. Rev. 44 (3) (2014) 87–95, [Online]. Available: https://doi.org/10.1145/2656877.2656890.
[124] P. Taffet, J.M. Mellor-Crummey, Lightweight, packet-centric monitoring of network traffic and congestion implemented in p4, in: 2019 IEEE Symposium on High-Performance Interconnects, HOTI 2019, Santa Clara, CA, USA, August 14-16, 2019, IEEE, 2019, pp. 54–58, [Online]. Available: https://doi.org/10.1109/HOTI.2019.00026.
[125] B. Turkovic, F.A. Kuipers, N.L.M. van Adrichem, K. Langendoen, Fast network congestion detection and avoidance using p4, in: D. Raychaudhuri, R. Li (Eds.), Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies, NEAT@SIGCOMM 2018, Budapest, Hungary, August 20, 2018, ACM, 2018, pp. 45–51, [Online]. Available: https://doi.org/10.1145/3229574.3229581.
[126] Y. Li, R. Miao, H.H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, M. Yu, HPCC: high precision congestion control, in: J. Wu, W. Hall (Eds.), Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019, ACM, 2019, pp. 44–58, [Online]. Available: https://doi.org/10.1145/3341302.3342085.
[127] F.Y. Yan, J. Ma, G.D. Hill, D. Raghavan, R.S. Wahby, P. Levis, K. Winstein, Pantheon: the training ground for internet congestion-control research, in: H.S. Gunawi, B. Reed (Eds.), 2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018, USENIX Association, 2018, pp. 731–743, [Online]. Available: https://www.usenix.org/conference/atc18/presentation/yan-francis.

Huiling Jiang received the B.S. degree (2019) from Tsinghua University, Beijing, China, in School of Economics and Management. She is currently a master at Tsinghua University, China. Her research interests include data center network, intelligent network, flow scheduling, congestion control, etc.

Qing Li received the B.S. degree (2008) from Dalian University of Technology, Dalian, China, the Ph.D. degree (2013) from Tsinghua University, Beijing, China; both in computer science and technology. He is currently an associate professor at Southern University of Science and Technology, China. His research interests include reliable and scalable routing of the Internet, software defined networks, network function virtualization, in-network caching/computing, intelligent self-running network, etc.

Yong Jiang received the B.S. degree (1998) and the Ph.D. degree (2002) from Tsinghua University, Beijing, China, both in computer science and technology. He is currently a full professor at the Graduate school at Shenzhen, Tsinghua University. His research interests include the future network architecture, the Internet QoS, software defined networks, network function virtualization, etc.

GengBiao Shen received the B.S. degree (2013) and the M.S degree (2016) from Beihang University, Beijing, China, both in instrument science and technology. He is currently a Ph.D. candidate at Tsinghua University, China. His research interests include data center network, in-network caching, intelligent network, software-defined networking, flow scheduling, load balancing, etc.

Richard Sinnott is currently a Professor from Melbourne University of Computing And Information Systems, Melbourne, Australia. He graduated from University of East Anglia and got his B.S in 1988. He graduated from University of Stirling and got his M.A degree and Ph.D. degree in 1993 and 1997, and worked as a director in University of Glasgow during 2002–2010. His Research interests include real time systems, clinical trials, distributed systems, etc.

Chen Tian received the B.S., M.S., and Ph.D. degrees from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China, in 2000, 2003, and 2008, respectively. From 2012 to 2013, he was a Post-Doctoral Researcher with the Department of Computer Science, Yale University. He was an Associate Professor with the School of ElectronicsInformation and Communications, Huazhong University of Science and Technology, China. He is currently an Associate Professor with the State Key Laboratory for Novel Software Technology, Nanjing University, China. Hisresearch interests include data center networks, network function virtualization, distributed systems, Internet streaming, and urban computing.

Mingwei Xu received the B.Sc. degree and the Ph.D. degrees from Tsinghua University. He is a full professor in Department of Computer Science and Technology at Tsinghua University. His research interests include computer network architecture, high-speed router architecture and network security.