ISSN: 2664-9640

APCC: Research on DRL based Adaptived Perceived Congestion Control Method for Information Center Networking

Yuejun Zhang [1, a], Mingchuan Zhang[1, b]

[1] School of Information Engineering, Henan University of Science and Technology, Luoyang, 471023, China.

[a] yjzhang@stu.haust.edu.cn, [b] zhang_mch@haust.edu.cn

Abstract

In recent years, the introduction of Information Center Network (ICN) has made up for the deficiency of traditional network architecture. The content-based network architecture replaces the original host-based network architecture and has the characteristics of multi-path transmission and node cache. However, with the rapid increase of network traffic and the increasingly complex transmission environment, the problem of network congestion has also been brought.To solve this problem, we propose an ICN Adaptive Perceptual Congestion Control mechanism (APCC) based on Deep Reinforcement Learning (DRL). In the stage of congestion control mechanism, we combine perception with decision, and drive and control the sending rate of interested pack-ets at the receiver according to the characteristics of the network. When the end user receives the packet and provides a return to the user, the user adjusts the rate of the interest packet for the next request based on the return. The APCC mechanism proposed in this paper is implemented in ICN, and compared with ICP and the latest DRL-CCP. The simulation results show that APCC has higher throughput performance, lower network latency and faster convergence speed, and can effectively avoid congestion.

Keywords

Information Centric Networking(ICN); Adaptive perceived; Congestion control; Deep Reinforcement learning.

1. Introduction

Nowadays, users' needs have changed from the focus on the location of the content to the focus on the requested content itself, and put forward diversified demands on the content itself. For example, real-time communication, online video [1], information retrieval, mul-timedia streaming and other high bandwidth application requests. These requirements pose some challenges to the traditional TCP/IP networkarchitecture,suchasmobility,caching,multi-pathtransport, and security. In order to meet these user requirements, we have made some modifications to the network architecture. Butatthesametime, it brings some performance problems in providing services to users. The lack of performance of the traditional network architecture, on one hand, causes the users can not get the requested content in time, on the other hand, also lays a foundation for the emergence of the new network architecture. Therefore, a new network architecture Information Center Network (ICN) [2] emerges in response to these problems.

With the emergence of Information Center Network architecture, the current host-centered network architecture is changed and the content-based network architecture is established. In ICN, content requests are mainly driven by the receiving end. Users only send their content requests through interest packets, search for matches based on content names through routing nodes, until the final con-tent storage node or terminal, and then transmit the requested packet

ISSN: 2664-9640

back in reverse according to the path. At present, many people have explored ICN network in different aspects, such as Named Data Network (NDN), Content Center Network (CCN) and so on [3–5]. Although ICN removes the relationship between content and host and simplifies the process of content delivery, it brings complexity to network congestion control due to the lack of an appropriate content-based request protocol. Therefore, how to use network traffic rationally to avoid network congestion is one of the challenges faced by ICN network at present. When multiple users access network resources (bandwidth, throughput, and delay, etc.) at the same time, how we provide a fair, efficient and reasonable resource sharing scheme for each user is very important. We found that traditional network TCP congestion control has been developed and mature in recent years, such as TCP Vegas [6], TCP New Reno [7], and BALIA[8], etc., but it cannot be directly used in ICN network, because ICN does not maintain end-to-end flow for each pair of communication hosts. In order to apply traditional network congestion control method, several flow-based congestion control mechanisms are proposed, but it is difficult to identify congestion on a single path due to the characteristics of ICN multi-node cache. And each traffic has a state, and maintaining the state of each traffic is more expensive in ICN. In order to eliminate the cost of maintaining state, a hope-by-hop congestion control method is proposed, but the same routing needs the support of ICN router, on the other hand, it cannot be applied to various network environments.

Therefore, considering the complexity of existing congestion control methods and the characteristics of ICN network, we proposed an adaptive perceptual congestion control (APCC) mechanism based

on Deep Reinforcement Learning (DRL). ICN interest and data exchange are driven by the receiver, and in the case of link overload, congestion control affects data requests at the receiver. Second, ICN's ubiquitous node cache [9] means that intermediate nodes can cache packets being passed in order to satisfy future user requests. These features add complexity to the design of congestion control protocols.InICN,contentispackagedandstoredondifferentrouters and servers. The complex and high-dimensional dynamic characteristics of ICN networks make it difficult to model, predict and control ICN networks.It needs to meet the different requirements of different nature, different distribution and different content. It is reasonable to say that the ideal congestion control mechanism of ICN should make full use of relevant information such as content information, network state and user demand to achieve optimal control, and adjust the congestion control strategy adaptively for dynamic scenarios.

The proposed congestion control mechanism is driven by the receiver, which adjusts the transmission rate of interest packets to avoid network congestion, and combines the characteristics of network with the perception and decision making in DRL. To perceive the network link state and determine the sending rate of interest packets without considering the network complexity caused by the cache in the node. The return function is further optimized by algorithm training and regarded as a utility maximization problem. When the payoff is larger, the transmission rate of interest packets can be increased, and the bandwidth can be used reasonably to avoid congestion.

In summary, we make the following contributions in this paper:

1)We propose an efficient and practical DRL-based ICN congestion control mechanism APCC, which uses pre-trained data and interacts with the current network environment to intelligently generate appropriate actions to adjust the interest packet transmission rate at the receiver.

2)We evaluated our solution through large-scale numerical simulations and prototype system experiments. Experimental results show that the proposed algorithm is superior to the existing solutions in improving link utilization paths with high throughput and low latency.

ISSN: 2664-9640

The rest of this paper is organized as follows: In Section 2, we first review the relevant work, and then introduce the DRL prior knowledge in Section 3. In section 4 introduces the design of APCC algorithm. In section 5, we evaluate the performance of our pro-posed algorithm on a simulation platform. In section 6 summarizes the thesis.

2. Related Work

Although the ICN was proposed by researchers several years ago, it is still in its early stages of development. Therefore, the existing congestion control algorithm cannot be directly applied to ICN network [10–13]. However, because of the connection-less nature of ICN, multiple data source, in-network caching and multi-path for-warding strategy in ICN routers, congestion control in ICN network is different from the current host-to-host IP architecture networks. Among the existing congestion control methods in the ICN network, we can divide the existing congestion control methods into three types: receiver-based congestion control mechanism, hop-by-hop expectation rate control mechanism and joint control mechanism.

2.1. Receiver-based Congestion Control Mechanism

There are two kinds of acceptance-driven congestion control: window-based and rate-based. The window-based congestion control method continues the end-to-end connection between hosts, and adopts the principle of additive increase multiplicative decrease (AIMD) similar to TCP protocol [14] to adjust the window size for congestion control. Carotiglio et al. [15] designed a window-based interest flow control protocol (ICP), which is driven by an additive multiplier reduction mechanism to adjust the receiver interest rate in CCN. On the basis of the existing storage sharing model, the protocol is analyzed and designed under the scenario of single and multi-bottleneck link through the interaction with the cache in the network, which ensures the optimal fair and effective bandwidth sharing. Ali et al. [16] proposed a Delay-based Congestion Control Protocol (DCP). The DCP is designed to detect and control congestion at the consumer end of the network based on ICN architecture, based on queue-delay estimation without the use of round-trip time measurements or additional congestion signals. Most window-based congestion control schemes combine the RCP [17] and the multi-path forwarding of ICN, and their advantages are to maximize the fair bandwidth and bandwidth utilization. Inspired by the RCP. Zhong et al. [18] proposed a rate-based, multi-path sensing congestion control algorithm (MNRCP). The transmission rate of interest packets is controlled by calculating the transmission rate of each stream and feeding it back to the consumer. Lei et al. [19] proposed a rate-based, explicit, hop-by-hop congestion control algorithm for NDN hop-by-hop RCP(NHBH-RCP)to achieve highlink utilization and improve overall network throughput. However, the receiver-driven approach is state fully, and maintaining the state of each stream imposes additional overhead on ICN communications.

2.2. Hope-by-hop Expectation Rate Control Mechanism

In order to reduce the cost of state maintenance, a hop-by-hop congestion control method is proposed in recent years. PCON [20] is a typical hop-by-hop congestion control scheme in which routers detect congestion on their local links by using the CoDel extended active queue management (AQM) scheme. When congestion is detected, a congestion signal is sent in advance and available band-width is considered. HoBHIS [21] detects and controls congestion on intermediate nodes by shaping interest to control interest sending rate, and adds a NACK feedback in HoBHIS to inform down-stream nodes of congestion. Natalya et al. [22] proposed a rate-based content-centric congestion control method, and one interest can only retrieve one data packet at most. Distributed calculation of the avail-able capacity of each router is used to dynamically adjust their data rate and transmission buffer occupation. And demonstrated the hop-by-hop performance. Wang et al.[23] proposed a hop-by-hop-based interest-shaping

ISSN: 2664-9640

congestion control method, which considers the gradual relationship between interest and data to formulate an optimization problem to achieve the optimal rate of interest-shaping method. Zhou et al. [24] proposed a control mechanism of active transmission interest rate Explicit Congestion Notifications (ECN). In order to obtain globally optimal data transmission, they further proposed a forwarding only mechanism to realize link utilization and packet loss rate of high-speed data transmission. However, hop-by-hop congestion control requires modification of the intermediate ICN router, which adds complexity to the ICN router.

2.3. Joint Control Mechanism

The joint congestion control mechanisms include HR-ICP [25], CHoPCoP [26], and EC-Cubic [27], etc. They detect the network congestion state of intermediate nodes and control the interest rate transmission rate of the receiver or intermediate nodes. However, in most cases, there is no proper collaboration between acceptor driven mechanisms and hop-by-hop mechanisms, which affects their performance. The feedback of network congestion detection and intermediate routing has a certain delay. The best congestion control method is to give users the ability to perceive and predict the state of the network. Sara et al. [28] proposed a mechanism to control the bandwidth of the shared network through concurrent flows to ensure satisfactory bandwidth sharing between each elastic stream to supplement the CCN network. Adrien et al.[29] proposed a cooperative congestion control based on cache, multi-path and multi-producer retrieval. The nodes cooperate with each other to supervise the output queue management multi-path capacity to achieve fair distribution of network traffic and to maximize user service quality. They are all based on the RTT mechanism [25][28]. When the RTT exceeds a predefined threshold, the link will be aware of congestion, which may cause severe network congestion. This suggestion can work in the stage of avoiding congestion, and by adjusting the inter-est packet, serious network congestion can be prevented in the early stage.

Compared with the existing congestion control solutions, we propose a congestion control mechanism through state awareness and adaptive link bandwidth, and select the optimal strategy to control the transmission rate of the requested interest packet by sensing the network state adaptive mechanism. On the one hand, the optimal strategy is chosen by sensing the end-to-end network state, which is consistent with the connectionless nature of ICN. On the other hand, without the participation of the intermediate router, the user can directly adjust the sending rate of the request interest packet according to the network state to reduce the complexity of the hop-by-hop congestion control.

3. Preliminaries

In this section, to make training algorithms easier to understand, we first provide some basic knowledge about Reinforcement Learning and a priori knowledge about replaying previous experiences.

3.1. Reinforcement Learning

We expressed an environment as a Markov Decision Process(MDP)[30–32] formalism for this work. An MDP is defined by a tuple (S, A, R, P, γ),which consists of a set of states S ,a set of actions A, a reward function $r(s,a)$, a transition function $p(s^{'} \mid s,a)$, and γ is the discount factor. Let $s(t)$ and $a(t)$ respectively denote the environmental state and action selected at time t. The strategy π specifies the actions that the agent will take for each state. In each $s \in S$,echo action $a \in A$. The agent's goal is to find the strategy π mapping state, which maximizes the expected discounted total reward in the agent's life cycle. We further denote K for the dimension of the action and state space, write akfor the kth component of an action $a \in A$ and

ISSN: 2664-9640

skfor the kth component of an state $s \in S$, that is, $a = (a_1, ..., a_K)$, $s = (s_1, ..., s_K)$ The goal of the agent is to find the policy π mapping states to actions that maximizes the expected discounted total reward over the agent's life time.

The definition is the long-term expected return of the strategy state s:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s(t), a(t)\right) | s(0) = s\right] \tag{1}$$

We will use the following standard definitions of the value function V^{π}, the state action value function Q_{π}.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma \max_{a}^{'} Q(s_{t+1}, a^{'}) | S_t = s, A_t = a\right] \tag{2}$$

The optimal value function $Q^{*}(s,a)$ provides maximal values in all states and is determined by solving the Bellman equation:

$$Q^{*}(s,a) = R_s^a + \gamma \sum_{s' \in A} p_{ss'}^a \max_{a'} Q^{*}(s^{'}, a^{'}) \tag{3}$$

For the finite state space and the action space, under the conditions of the continuous state space and the action space, there is a deterministic optimal strategy.

DQN [33] adopts the convolutional neural network to approximate the action value function. One uses the target Q network to pdate the target, and the other uses experience playback. First use a separate target network, let the network replicate τ step from the regular network, so that the target Q-value will be more stable. The agent then adds all of its experiences to the replay buffer D^{replay} for playback, and then samples the replay buffer evenly to perform updates on the network to generate new policies.

3.2. Prioritized Experience Replay

Experience playback enables [34] online reinforcement learning agents to remember and reuse past experiences. In previous studies, past experiences (transition, a record in the experience pool, represented as meta-ancestor form, including state, action, reward, discount factor, and next state) are only obtained through uniform sampling. However, this method, as long as the previous experience,

then with the same probability as other experience will be reused, ignoring the importance of each of these experiences.

Random sampling is introduced to interpolate between the pure greed priority and the uniform random sampling, so that the sampled probability is monotonous on the priority of the transfer, and at the same time, the probability of even the lowest priority transfer is non-zero. We define the sampling transition probability x to be, $g(x) = \frac{g_x^{\mu}}{\sum_k g_k^{\mu}}$. Proportional prioritization where $g_x = |\zeta_x| + \varepsilon$, and ζ_x is the last TD (Temporal Difference) [35] error calculated for this transition and ε is a small positive constant to ensure all transitions are sampled with some probability. Priority playback introduces bias because it changes the data distribution, changes the expected value. The authors modified the weights by means of importance sampling, $m_i = (\frac{1}{N}, \frac{1}{g_i})^{\eta}$, N s memory sizes and η controls the amount of importance sampling with no importance sampling when $\eta = 0$ and full importance sampling when $\eta = 1$. η is annealed linearly from η_0 to 1.

ISSN: 2664-9640

4. Congestion Control based on DRL

If you follow the "checklist" your paper will conform to the requirements of the publisher and facilitate a problem-free publication process. In this section, we describe in detail our proposed Adaptive Perceptual Congestion Control (APCC) algorithm. We first give an overview of congestion control and describe the motivations and objectives for improving congestion control. This paper introduces how to control the rate of interest packets sent by the receiver driver by deep reinforcement learning method, and defines the input state and return function of the algorithm. Finally, the detailed process of the training algorithm is elaborated, and the algorithm is summarized.

4.1. Overview of Congestion Control

Congestion refers to the phenomenon of congestion caused by the insufficient processing of the requested service by the network link. More serious will lead to the breakdown of the entire network and the decline of network performance. The congestion control is a response made according to the network state to avoid congestion, improve network utilization and reduce delay and packet loss. There are many existing network congestion control methods, but the scope of consideration is not comprehensive. Therefore, we propose an adaptive sensing congestion control method based on DRL. By adjusting the sending rate of user interest request, the state of the adaptive sensing link can generate the optimal strategy and select the appropriate sending rate. In this way, the link utilization in the data transmission stage is improved and the network congestion phenomenon is avoided.

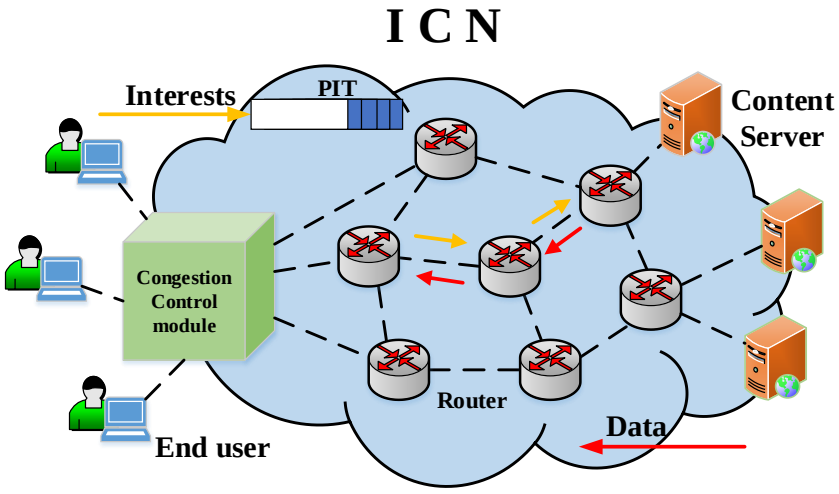


Figure 1. Congestion control mechanism deployed in ICN

4.1.1 Motivation

As a new network architecture, ICN makes up for the deficiency of the traditional network and changes the end-to-end transmission mode of the traditional network. It uses the node cache to replace the terminal server cache, making it more convenient for users to obtain content. But at the same time, node caching and multi-path transmission undoubtedly increase the complexity of ICN network. The proliferation of network users and the variety of content requests put forward the requirements for the response speed of the link. When the link response is not timely, it will cause serious link congestion, or it will lead to network breakdown. Therefore, in order to solve the congestion problem in ICN, a congestion control method suitable for ICN is proposed. The traditional congestion control method is to reduce the size of the control window by addition and multiplication according to the ACK returned, which cannot be applied to ICN, and the characteristics of ICN need to be considered. The existing ICN

ISSN: 2664-9640

congestion control method is improved on the basis of the traditional network congestion control, and it is only controlled by the feedback information of nodes, which increases the complexity of congestion control. Therefore, we propose an adaptive sensing congestion control method, which treats the congestion control problem as a utility maximization problem, and uses deep reinforcement learning to train the perceived network state only by selecting the optimal strategy for congestion control. As can be seen from Figure 1, we deploy the training device in the receiver driver and control the sending rate of the interest packet of the receiver driver through the generated decision.

4.1.2 Control Procedure of APCC

In ICN, when a user sends an interest packet request, the size of the interest packet is negligible relative to the size of the returned packet, and the link bandwidth is actually consumed by the returned packet. Therefore, we separate the DRL training part from the interest pack request and receive packets and extract data from them as training data. Firstly, a training model based on DRL and related variable parameters is designed. Next, in the training part, we use the existing data in the network to pre-train them, and use the database to store the sampled data in the experience playback, and then interact with the real network environment to generate new strategies. Finally, in the transport phase, we control the rate at which the consumer sends packets of interest across the link. The condition of the network environment determines how to select and update the strategy of the training part. The former selects the policy according to the link bandwidth and delay time, while the latter selects the policy according to the network status update policy.

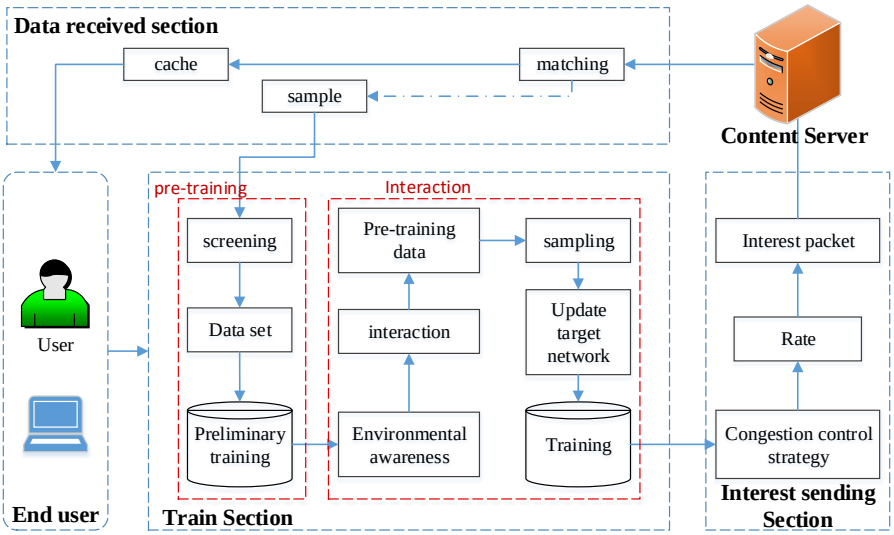


Figure 2. Implementation of APCC mechanism

From Figure 2, we can see that ICN network is mainly divided into four parts: users, congestion control, interest transmission and data reception. Part of congestion control module is the training we will it is divided into two parts one is the preliminary stage of training data is an interactive online learning stage, when users send interest packet request need through the congestion control module on the network environment perception (link bandwidth, delay, packet loss) filtered through the previous network data sampling to generate a database in playback experience carries on the preliminary training, through training the generated data with real network environment interact through online learning to generate new data storage to experience the playback according to network state to generate the corresponding decision. The interest sending part controls the sending rate of interest packets according to the generated decision control. In the receiving stage, we update the network state through the

ISSN: 2664-9640

network information carried by the received packets to provide policies for the next user request.

4.1.3 DRL-based Congestion Control

Deep reinforcement learning combines both deep learning and reinforcement learning to coexist perception and decision making. One is to learn from past experiences and the other is to generate strategies based on learned experiences. Deep reinforcement learning is applied to ICN to learn the existing network state and generate the optimal strategy to adapt to the current network link bandwidth according to the learning experience. In the face of complex high-latitude state space like ICN, we need to manually select state space and actions as training targets to reduce computational complexity and convergence speed. We proposed an adaptive congestion control perception method by using markov decision process model, the depth DQfD in reinforcement learning algorithm is adopted to obtain the data information of training, the depth of the compared with other reinforcement learning algorithm, more applicable to real network, and use a small amount of data samples can generate optimal strategy method. Therefore, we will describe the selection of state spaces and the process of how to generate decisions from these training data.

4.1.4 Object

Our goal of congestion control is to improve network performance by maximizing link utilization while avoiding network congestion, and to generate different congestion control strategies adaptively for different content requests. We think of the congestion control problem as a utility maximization problem, where each strategy has a corresponding payoff. Obtain the network status according to the return and adjust the sending rate according to it to improve the user's interest satisfaction rate of request.

4.2. State

The state information represents the environmental information perceived by the agent and the changes brought about by the agent's own behavior. Country information is the basis for agents to make decisions and evaluate their long-term interests. By combining perception with decision making, DRL can solve the problems brought about by continuous and complex high-dimensional state space, but at the same time, we need to consider the complexity of training and the convergence speed of algorithm. In the real network environment, there are many parameters that affect the network state, and the state selection directly determines whether the DRL algorithm converges, the rate of convergence and the final performance. Therefore, it is necessary to optimize the state space. In order to address computational complexity and convergence speed, we have artificially selected a number of variables as criteria for the state space to maximize performance. We refer to the time interval between a user sending a packet of interest and receiving the corresponding packet. Here, we define the state space as an 8-dimensional variable. Therefore, in order to achieve good performance, it is necessary to select the appropriate input state for the network state. The state vector is see Table 1.

Table 1. State vector

Variable	Definition
c_{time}	the connection bandwidth at time t
i_{time}	the interest packet sending rate at time t
i_{count}	the total number of interest messages sent during the monitored interval
d_{count}	the total number of packets received during the monitoring interval
l_{count}	total number of packets retransmission during the monitor interval
i_{size}	average size of Insterest packets received during the monitor interval

ISSN: 2664-9640

d_{size}	average size of Data packets received during the monitor interval
d_{rtt}	average RTT of Data packets received during the monitor interval

4.3. Action

In order to accurately control the transmission rate of interest packets, it is very important to select appropriate actions according to the returns in the algorithm training stage. In the design of APCC, considering the dynamic and changeable network, we should change the transmission speed of user interest packets according to the different state of the network to respond to the change of the current network environment. Considering the convergence speed and computational complexity of the algorithm, we can choose three actions to simplify the action space based on the DRL model see Table 2.

Table 2. Action of APCC

Actions	Change size(Mbps)
Increase	1
No change	0
Decrease	1

These actions directly affect the sending rate of interest packets requested by the receiver, and the user changes the sending rate of interest packets according to the state of the network. When the network is in good condition we speed up our sending rate according to the return from the previous action, when the return is low we reduce our sending rate, and when the return is the same, the rate remains the same. When the user selects an action based on the state of the network, it remains unchanged until the previous action is completed.

4.4. Reward

The goal of APCC is to maximize returns, so the definition of the reward function should be consistent with the objectives of other congestion control algorithms. Think of it as an optimization goal [36, 37] in the congestion control phase, our goal is to make full use of bandwidth while avoiding network congestion. We should reward throughput and punish network congestion. We consider that throughput, delay and packet loss are more representative signals of link congestion state. Therefore, the key idea of designing reward functions is to make full use of bandwidth while avoiding network congestion. At the same time, we should punish network congestion while rewarding throughput. We do not consider retransmission because when the user does not receive the corresponding packet, there will be a new request. In the case of poor link state, retransmission of data will cause link congestion and lead to unbalanced network load.

The definition of reward is important to APCC. We design the single content's utility function as follow:

$$\begin{aligned} Utility_i(t) = \alpha_i \times \log_i(throughput_i(t)) \\ -\gamma_i \times RTT_i(t) - \beta_i \times loss_i(t) \end{aligned} \tag{4}$$

when the user requests a packet of interest, we select the next action size through our return function, which includes throughput, delay, and loss rate, which are α , γ and β are the important parameters affecting performance. The log function of throughput ensures that multiple content requests converge while sharing links. The control goal of a single content

ISSN: 2664-9640

request is to maximize throughput while minimizing latency and loss rates. When the return on a single request is maximum, the global return will be optimal and network performance will be improved by increasing network utilization.

Most congestion control methods do not give different policies based on what the user requests, but give the same transport policy based on the state of the network. However, with the continuous increase of network big data flow requests and the global outbreak of 2019-nCoV, people are forced to understand the real-time situation trend only through the media. Most of these real-time requests are video communication and network video requests. For such large video stream requests, we should not only avoid packet loss and delay in the network state, but also ensure the timeliness of the video request. Therefore, we define a separate utility function for time-sensitive user requests that takes effect on the return of the video stream request:

$$R_i^{video} = \frac{1}{N}\sum_{i}^{n} Utility_i(t) \tag{5}$$

This R_i^{video} stands for the reward of the user requesting the video.

Maximizing the total utility function is the ultimate congestion control goal in ICN network. By making full use of network bandwidth resources through environmental awareness, the optimal strategy is made to maximize the return. At the same time, the bandwidth is fully utilized to meet the needs of users who do not need to request and make decisions. As an important return parameter, the action selected in the current state will be rewarded according to the time of the packet received by the user. This reward is an important reference in the next network state. We use Deep Reinforcement Learning to train it, so as to maximize its long-term utility.

4.5. Training Algorithm

We consider both the convergence rate and computational complexity of the algorithm according to the computing power of the device and the problems it faces. We introduce Deep Q-learning from Demonstrations (DQfD) model \cite{b38} in deep reinforcement learning and propose the effectiveness of the mechanism. We identified targets for control and model the congestion control as a Markov Decision Process (MDP), We define (S, A, P, R, γ).

These elements are described in detail below. The Q-value of each state-action combination (s, a) is $Q(s, a)$, which reflects the quality of performing action a when the environment is in state s. The Q-values are updated every time an action a is taken in a state s, resulting in a reward $r = R_a(s, s')$ and a new state $s' \in S$. So in this case, we can make the reward $r = Utiliy(t)$ in each monitor interval.

The real network environment is complex and changeable, so most of the Deep Reinforcement Learning cannot be fully applied to the real network environment, such as self-driving cars, data centers, and high-speed networks, etc. These algorithms usually need a large number of data sets and thousands of training to achieve good performance in real situations. Therefore, in the face of complex, high-dimensional and changeable ICN network environment, what method should be adopted to solve the congestion problem caused by the surge of network traffic, and improve the network property and improve the network utilization rate to avoid the generation of congestion.

Therefore, we propose an adaptive perceptual congestion control (APCC) method and divide it into two phases when dealing with the congestion control of high latitude state space network. In the first stage, we select the parameters in the current network by sampling method, and train them through some loss functions. In the second stage, we interact the pre-trained strategies with the real network environment to generate new strategies. We will update the network state by mixing both pre-trained data and self-generated data.

ISSN: 2664-9640

In the algorithm pre-training stage, we sample and screen the existing network data, such as bandwidth, delay and packet loss, which are very important for network interaction after pre-training. Therefore, we use time difference (TD) and supervised loss to pre-train the pre-extracted data for the existing data. The supervised loss function enables the algorithm to learn the current network state, while the TD loss function enables the algorithm to learn the corresponding value function and continue to learn. After the initial training process, the learning experience accelerates the generation of strategies in the playback buffer, and then generates new strategies to interact with the existing network state and update the network state.

In the pre-training phase, we use four loss functions to train and update network parameters. The four loss functions are 1-step double-q learning loss, n-step double-q learning loss, supervision loss, and L2 regularization loss of network weights and expectations. The supervised loss function is used to classify the behavior of the network state output. Q-loss is to ensure that the network satisfies the Bellman equation, which can better act on the TD loss of the second stage. With the loss function, we can approximate the network, learn what we have learned more quickly, and then follow good generation strategies when applying it to real scenarios.

Algorithm 1 Adaptive Perceived Congestion Control Algorithm.

1: Input: $\mathcal{D}^{replay}$: initialize the pre-training data set, s_0 : Initialize network state, δ : current network weight (random), δ_0 : target network weight(random), τ : frequency at which to update target net, k : number of pre-training gradient updates

2: for steps $t \in \{1,2,...,k\}$ do

3: Sample of n transitions from $\mathcal{D}^{replay}$ with prioritization

4: Calculate loss $\Phi(Q)$ using target network

5: Perform a gradient descent step to update S

6: **if** $t \bmod \tau = 0$ **then** $\delta \leftarrow \delta'$ **end if**

7: **end for**

8: **for** steps $t \in \{1,2,...,k\}$ **do**

9: Take training samples from the movements $a \sim \pi^{\varepsilon Q_\delta}$

10: Update policy by one step of gradient ascent using

11: Play action a and observe (s', r) .

12: Store (s, a, r, s') into $\mathcal{D}^{replay}$, If the capacity is too large, the new data directly overwrites the old data

13: Sample of n transitions from $\mathcal{D}^{replay}$ with prioritization

14: Calculate loss $\Phi(Q)$ using target network

15: Update δ with gradient descent

16: **if** $t \bmod \tau = 0$ **then** $\delta \leftarrow \delta'$ **end if**

17: $s \leftarrow s'$

18: **end for**

Monitoring loss in the training process is the most important link of network training. Since we only extract a small amount of network state space, these small amounts of data cannot

ISSN: 2664-9640

generate better performance strategies, so many generated policies cannot directly adapt to the real network environment. Therefore, if we only pre-train the network, the state update intelligence of the next network depends on Q-learning to update, and the network state intelligence updates according to the Q-value, which will be transmitted to the whole network. Therefore, we add a maximum marginal classification loss function:

$$\Phi_y(Q) = \max_{a \in A}\left[Q(s,a) + j(a_y,a)\right] - Q(s,a_y) \tag{6}$$

where, a_y is the action generated in the state of s after interacting with the real environment, and $j(a_y,a)$ is the boundary function, which is the action with the greatest return (when $a = a_y$, it is 0, otherwise it is positive value). We use loss function for some action value function is lower than these actions make the reasonable boundary value function, otherwise these behaviors under continuous high network status space will be no constraint, make q network cannot satisfy the bellman equation, so we in the second stage through the interaction with the real environment and use the update TD network and modify its strategy.

The disadvantage of using One-Step is that the policies generated in the current state only affect the current state behavior values; the other state policy value pairs are updated using the Q value. Therefore, this affects the network learning rate, so we need to make more updates to pass the Q value to the associated status action reward. And we add an n-step return value to help the strategy generated in the network pre-training stage to better propagate to the previous network state and be better used for pre-training. The n-step return [39] is as follows:

$$r_t + \gamma r_{t+1} + \ldots + \max_a \gamma^n Q(s_{t+n}, a), \tag{7}$$

the single reward r image has n state action value pairs, and the propagation process is more effective.

We also add the weight of bias and the loss of L2 regularization applied to the real network environment to help the situation of the relatively small over-fitting of the pre-training data set in the training data set in the real network environment. The following is the combination of loss functions used to update the network:

$$\Phi(Q) = \Phi_{DQ}(Q) + \lambda_1\Phi_n(Q) + \lambda_2\Phi_y(Q) + \lambda_3\Phi_{L2}(Q) \tag{8}$$

the λ parameter controls the weight of these losses.

Once the pre-training phase is completed, ideally the agent has learned a sound strategy so that it can run reliably on a real system. In the next phase, the agent starts running on the real network, collecting the data it generates, and adding it to the replay buffer. New data is added to the reset cache until the buffer cache is filled up and the old data is overwritten with new data. In the meantime, the pre-demo data remains in a separate demo playback buffer and remains unchanged.

The APCC pseudo code is in Algorithm 1. The behavior policy $a \sim \pi^{\varepsilon Q_\delta}$ is ε-greedy with respect to Q_δ.

5. Simulation Results and Discussions

In this section, we will introduce the experimental scenarios and parameter settings. The numerical results of the proposed dispersion algorithm are also analyzed. In the end, we evaluate the performance of APCC in some typical ICN transmission scenarios and compare it with two congestion control algorithms ICP [15] and DRL-CCP [40]. The recovery rate, time delay and packet loss rate are compared.

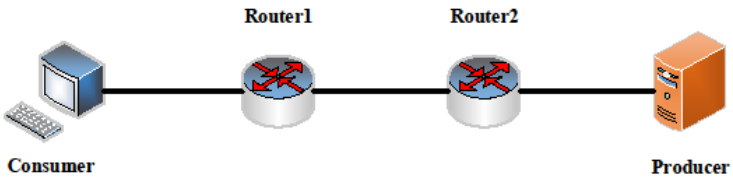
ISSN: 2664-9640

5.1. Experiment Setup

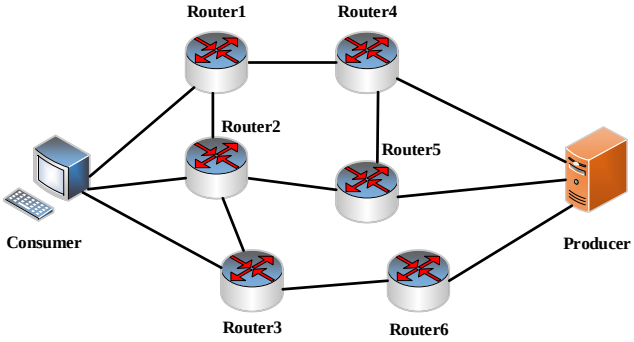
The convergence and optimality of APCC algorithm are verified by a series of numerical simulations, to sum up, APCC can achieve fairness and good network utilization.

In order to evaluate the performance of our proposed APCC algorithm, our experiment was carried out on a topological model. Figure 3 shows us the severe network architecture. According to the requirements of different scenarios, different network architectures are generated to set different link bandwidths.

This paper introduces the DQfD algorithm. The parameters used in the experiment are summarized in the table. Through proper parameter configuration, the optimal congestion control strategy is finally obtained after training.



(a)



(b)

Figure 3. Network topology

5.2. Scenarios and Baseline

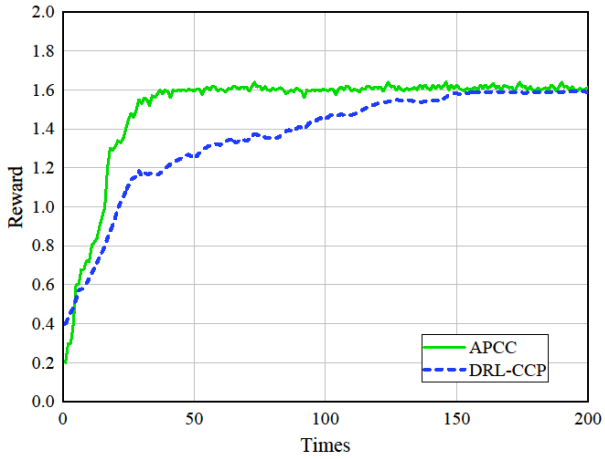


Figure 4. Performance analysis of algorithm in real network

The payback functions of different algorithms in the same network topology are shown in Fig. 4. It can be seen that under the same network architecture, APCC algorithm proposed by us, compared with DRL-CCP, has a higher return and faster convergence rate after 10,000

ISSN: 2664-9640

iterations of its return function. As it gets a better sense of the state of the network and interacts with the environment with each user request, it continues to sample and learn from its own generated and demo data. In each small batch, the ratio of these two types of data is automatically controlled by the priority replay mechanism. However, although DRL-CCP can also carry out good learning, each time it needs to be trained through a large amount of data, the learning time is long, the convergence speed is slow, and its return is not as high as that of APCC.

5.3. Scenarios and Performance

In our experiment, we studied the performance of APCC under different ICN network architectures, mainly considering the performance under different environments from link utilization, delay and packet loss. Here link utilization is obtained by multiplying the rate at which the user requests interest packets multiplied by the ratio of the size of the packets to the size of the interest packets. Delay is measured by the time it takes for a user to request a packet of interest to receive the packet, while packet loss is measured by the time the request is not received.

5.1.1 Fixed Bandwidth Networks

We conducted experiments in the multi-path network topology, as shown in Figure 3(b), where the bandwidth was set to 75Mbps (packet size 1-25kbits was randomly requested) and the delay of each link was 2ms.

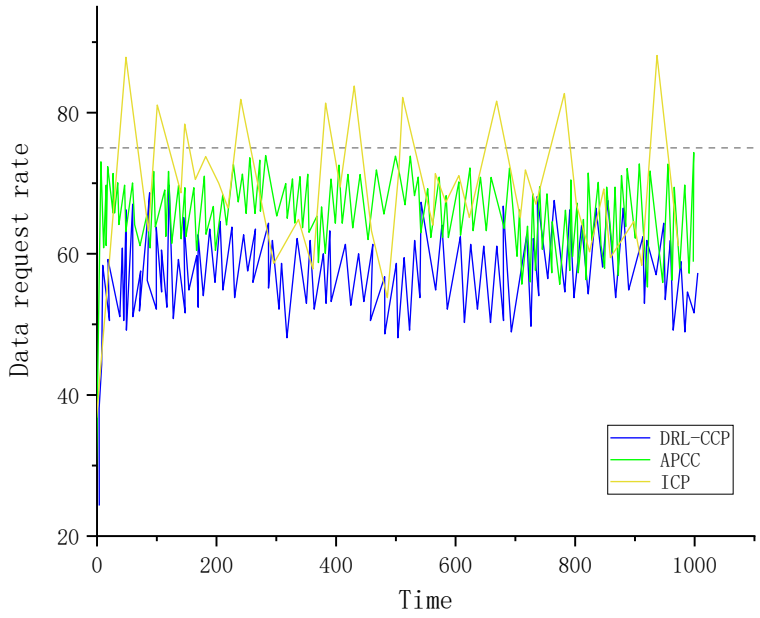


Figure 5. ICN network link utilization

The simulation results are shown in Figure 5, including three different methods in fixed bandwidth and interest request rate and delay box graphs.As can be seen from the figure, the adaptive sensing APCC method proposed by us can make full use of the link bandwidth set by us, and is more stable and has better performance compared with the other two DRL-CCP and ICP methods. Because ICP uses AIMD (increase by addition and decrease by multiplication) to control the window size, it is easy to exceed the set bandwidth value when the user requests data at the beginning, resulting in link congestion and network delay. Then, according to the congestion control request timeout, the user request has no response and the user request rate is halved, resulting in the loss of the request.Although in the network initialization phase ICP can by sending a request to test the network bandwidth, but need through the training for

ISSN: 2664-9640

thousands of times, long training time and cannot be adapted to the real network environment, cannot provide good for ICN network congestion control method, especially in the real network in which DRL-CCP is very difficult to to estimate delay and packet loss. Therefore, the APCC algorithm proposed by us can continuously learn from experience and interact with the environment, so that the transmission rate can be adjusted independently in the real network environment. As we can see from the diagram, APCC can achieve a stable state in the initial stage and maximize utilization with low delay and packet loss rate.

5.1.2 Packet loss

As can be seen from Figure 6, with the increase in the number of network user requests, more packets were lost at the beginning of training.As the training time increases, the packet loss rate of APCC algorithm begins to decrease and flattens out to only about 2%.However, the other two algorithms have different packet loss problems and can't control packet loss well.Therefore, compared with DRL-CCP and ICP algorithms, our proposed adaptive sensing method has the lowest packet loss rate and lower retransmission probability.

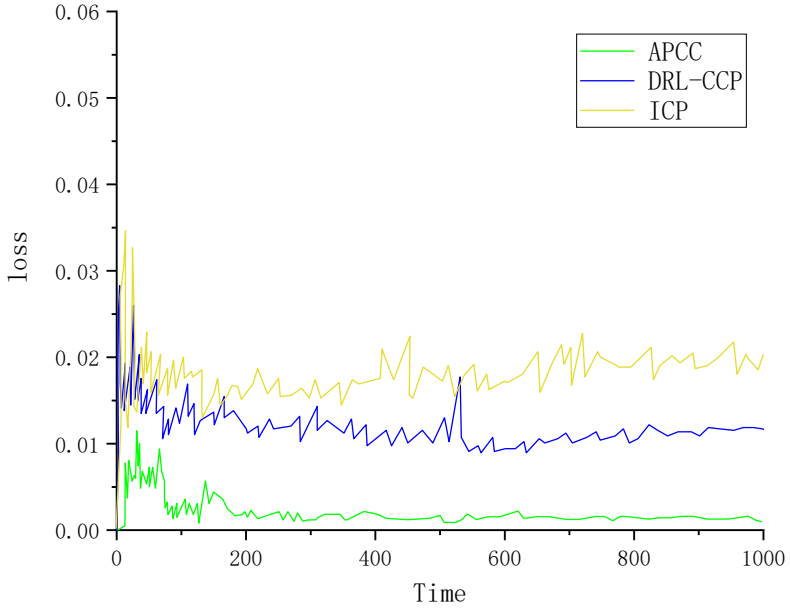


Figure 6. Packet loss rate

5.1.3 Time delay

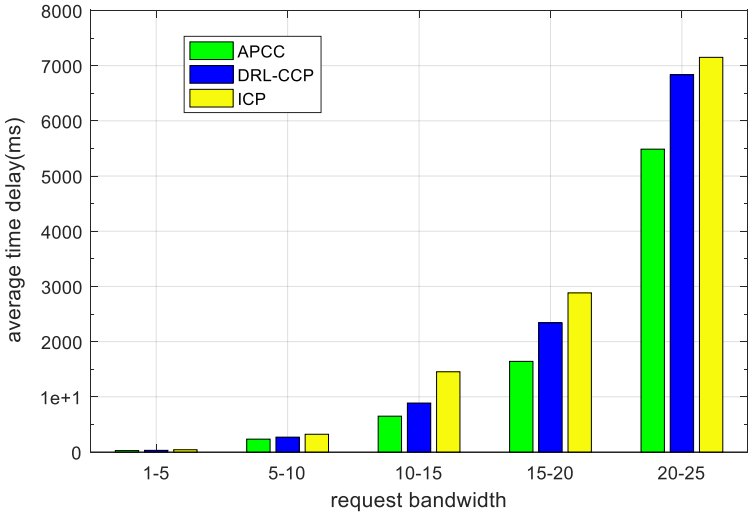


Figure 7. Average time delay

ISSN: 2664-9640

As can be seen from Figure 7, the request rates are classified. With the increase of data requests, the bandwidth is fully utilized while the network latency is also increasing. Compared with other algorithms, our algorithm has the lowest latency.

6. Conclusion

Based on the ICN network architecture, this paper proposes a receiver-driven congestion control mechanism, APCC. In this mechanism, the ICN receiver selects a strategy for congestion control according to the network state. The receiving end evaluates the current network environment (bandwidth, delay, throughput) according to the perceived network state, and then adjusts the transmission rate of the interest packet request according to the network state, that is, returns. The adjustment of the transmission rate is viewed as a utility maximization problem, which maximizes the long-term benefits to the user. A DQfD algorithm based on deep reinforcement learning is proposed. We implement APCC mechanism on simulation platform and test its fairness under different network bandwidths. A large number of experiments show that compared with advanced ICN congestion control methods, APCC has higher throughput, shortest latency and faster convergence speed.

Despite these advantages, we also discussed some limitations and possible future development directions. In ICN, different content has different popularity. Nodes tend to cache the content with high popularity and share the content with other nodes to other nodes. Therefore, more accurate classification according to the content popularity of different business streams can better control the cost, and can obtain the desired data faster during future visits. Therefore, we plan to distinguish the content of user requests, and combine the congestion control mechanism with the intra-node cache to jointly optimize and better avoid congestion control.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. U1604155.

References

[1] M.Zhang , B.Hao , F.Song, et al. Smart collaborative video caching for energy efficiency in cognitive Content Centric Networks[J]. Journal of Network and Computer Applications, 2020, 158:102587.

[2] A V .Vasilakos , Z.Li , G.Simon , et al. Information centric network: Research challenges and opportunities[J]. Journal of Network and Computer Applications, 2015, 52(jun.):1-10.

[3] L. Zhang , A. Afanasyev , J. Burke , et al. Named data networking[J]. Acm Sigcomm Computer Communication Review, 2014, 44(3):66-73.

[4] G. Xylomenos . A survey of information-centric networking research[J]. IEEE Communications Surveys & Tutorials, 2013, PP(99):1-26.

[5] S H. Ahmed , D. Kim , S H. Bouk . Content-Centric Networks: An Overview, Applications and Research Challenges[M]. Springer Publishing Company, Incorporated, 2016.

[6] S W. O'Malley , L S. Brakmo , L L. Peterson . TCP Vegas: new techniques for congestion detection and avoidance[J]. ACM SIGCOMM Computer Communication Review, 1994, 24(4).

[7] S. Floyd , A. Gurtov , T. Henderson . The NewReno Modification to TCP's Fast Recovery Algorithm. 2004.

[8] Q. Peng , A. Walid , J. Hwang , et al. Multipath TCP: Analysis, Design and Implementation[J]. 2013.

[9] M . Zhang , B. Hao , R. Wang , et al. A Pre-Caching Strategy Based on the Content Relevance of Smart Device's Request in Information-Centric IoT[J]. IEEE Access, 2020, PP(99):1-1.

[10] F .Song , Z. Ai , Y. Zhou , et al. Smart Collaborative Automation for Receive Buffer Control in Multipath Industrial Networks[J]. IEEE Transactions on Industrial Informatics, 2019, PP(99):1-1.

[11] V. Jacobson , D K. Smetters , J D. Thornton , et al. Networking named content[J]. Communications of the ACM, 2012.

[12] Ahlgren, et al. "A survey of information-centric networking." IEEE Communications Magazine Articles News & Events of Interest to Communications Engineers (2012).

[13] G. Xylomenos . A survey of information-centric networking research[J]. IEEE Communications Surveys & Tutorials, 2013, PP(99):1-26.

[14] Jacobson, V. Congestion avoidance and control[J]. Acm Sigcomm Computer Communication Review, 1988, 18(4):314-329.

[15] G. Carofiglio , M. Gallo , L. Muscariello . ICP: Design and evaluation of an Interest control protocol for content-centric networking[J]. Colloids and Surfaces A: Physicochemical and Engineering Aspects, 2012, 395:10-17.

[16] A A. Albalawi, J J. Garcia-Luna-Aceves . A Delay-Based Congestion-Control Protocol for Information-Centric Networks[C]// 2019 International Conference on Computing, Networking and Communications (ICNC). 2019.

[17] N. Dukkipati . Rate Control Protocol (RCP): Congestion control to make flows complete quickly. These Instructions, 2007.

[18] S. Zhong , Y. Liu , J. Li , et al. A Rate-Based Multipath-Aware Congestion Control Mechanism in Named Data Networking[C]// 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). IEEE, 2017.

[19] S. Xing , B. Yin , J. Yao , et al. A VCP-based Congestion Control Algorithm in Named Data Networking[C]// 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, 2018.

[20] K. Schneider , C. Yi , B. Zhang , et al. A Practical Congestion Control Scheme for Named Data Networking. 2016.

[21] N. Rozhnova , S. Fdida . An extended Hop-by-hop interest shaping mechanism for content-centric networking[C]// GLOBECOM. 2014.

[22] N. Rozhnova , S. Fdida . An effective hop-by-hop Interest shaping mechanism for CCN communications[C]// 2012 Proceedings IEEE INFOCOM Workshops. IEEE, 2012.

[23] Y. Wang , N. Rozhnova , A. Narayanan , et al. An improved hop-by-hop interest shaper for congestion control in named data networking[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4).

[24] J. Zhou , Q. Wu , Z. Li , et al. A proactive transport mechanism with Explicit Congestion Notification for NDN[C]// IEEE International Conference on Communications. IEEE, 2015.

[25] G. Carofiglio , M. Gallo , L. Muscariello . Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks[J]. ACM SIGCOMM Computer Communication Review, 2012.

[26] F. Zhang , Y. Zhang , A. Reznik , et al. A transport protocol for content-centric networking with explicit congestion control[C]// International Conference on Computer Communication & Networks. IEEE, 2014.

[27] Y. Liu , X. Piao , C. Hou , et al. A CUBIC-Based Explicit Congestion Control Mechanism in Named Data Networking[C]// International Conference on Cyber-enabled Distributed Computing & Knowledge Discovery. IEEE, 2017.

[28] S. Oueslati , J. Roberts , N. Sbihi . Flow-Aware traffic control for a content-centric network,[J]. IEEE, 2015.

[29] A. Thibaud , J. Fasson , F. Arnal , et al. Cooperative Congestion Control in NDN[C]// ICC 2020. 2020.

[30] Peter Dayan. Reinforcement learning. Stevens'Handbook of Experimental Psychology, 2002.

[31] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning, volume 135. MIT press Cambridge, 1998.

[32] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018

ISSN: 2664-9640

[33] M. Volodymyr , K. Koray , S. David , et al. Human-level control through deep reinforcement learning[J]. Nature, 2019.

[34] T. Schaul , J. Quan , I. Antonoglou , et al. Prioritized Experience Replay[J]. Computer Science, 2015.

[35] R. Sutton , H. Maei . Temporal-Difference Learning. MIT Press, 2007.

[36] Y. Zhou , M. Zhang , J. Zhu , et al. A Randomized Block-Coordinate Adam online learning optimization algorithm[J]. Neural Computing and Applications, 2020.

[37] C. Xu , J. Zhu , D O. Wu . Decentralized Online Learning Methods Based on Weight-Balancing Over Time-Varying Digraphs[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2018:1-13.

[38] T. Hester , M. Vecerik , O. Pietquin , et al. Deep Q-learning from Demonstrations[J]. 2017.

[39] G. Hu , C. Wu . Incremental multi-step R-learning[J]. Journal of Beijing Institute of Technology (English Edition), 1999, 8(3).

[40] D Lan, Tan X , Lv J , et al. A Deep Reinforcement Learning Based Congestion Control Mechanism for NDN[C]// ICC 2019 - 2019 IEEE International Conference on Communications (ICC). IEEE, 2019.