Edge Computing Aided Congestion Control using Neuro-Dynamic Programming in NDN

Jin Qin*, Yitao Xing*, Wenjia Wei†, Kaiping Xue*†‡
*School of Cybersecurity, University of Science and Technology of China, Hefei 230027, China
†Department of EEIS, University of Science and Technology of China, Hefei 230027, China
‡Corresponding Author, kpxue@ustc.edu.cn

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322365

***Abstract*—Named data networking (NDN) is an emerging network paradigm that decouples content from its storage location by providing one or more content copies and distributing them within the whole network. Congestion control is a fundamental and important problem in NDN, but it has not been well solved yet. Existing works can be divided into three main types, receiver driven flow based control, hop-by-hop interest shaping and hybrid control. While they are faced with more or less high computational complexity, multi-content source and multi-transmission path problems, we proposed our edge computing aided congestion control scheme (EACC). The main idea is to detect congestion along the transmission path and avoid it by interest forwarding control at edge nodes. We add a new field to data packet to record the congestion status of the transmission path when it returns. After that, we deploy the core computing functions of the solution at edge nodes, and formulate the interest packet forwarding control into a local MDP (Markov Decision Process) problem based on the returned path congestion status and local user request information. Then we use neuro-dynamic programming (NDP) to solve this decision problem and present a practical implementation at edge nodes. The proposed scheme is implemented in ndnSIM simulator and compared to other two methods. Simulation results show the effectiveness of our scheme.**

***Index Terms*—Named-Data Networking (NDN), congestion control, Markov Decision Process (MDP), neuro-dynamic programming (NDP)**

I. Introduction

With the development of network technology and user demands, information content services represented by online videos and social networks have dominated Internet services. Traditional TCP/IP networks is a host-centric communication architecture. When faced with these new network applications, it often encounters challenges such as redundant transmission, multi-path transmission, and user mobility. Information centric networking (ICN) is an emerging network paradigm that decouples content from its storage location by providing one or more content copies and distributing them within the whole network. Contents can be cached on any device at any location to meet the same request received in the future and reduce redundant transmission in the network.

Named data networking (NDN) [1] is a promising implementation of ICN which follows the same basic philosophy as it is a content centric architecture. Content acquisition in NDN is based on an user-driven publish/subscribe model, whereby users' requests may be satisfied by any network device. Interest packets and data packets are the two basic types of data packets in NDN and they have an one-to-one correspondence. At the same time, due to the content-centric data acquisition and name-based forwarding control, the transmission in NDN is quite different from that in traditional TCP/IP networks. Receivers can get content from the origin server or any intermediate node that has the cached content and requests can be dynamically multi-path forwarded because of the separation of routing and forwarding in NDN.

Due to these new features of transmission mentioned above, traditional TCP congestion control [2] cannot be directly applied to NDN. Different content sources and paths have different RTTs, making RTO-based congestion detection unreliable and the rate control mechanism based on single congestion control window no longer applicable. In addition, the data packets in NDN transmission are not necessarily ordered, which also makes the congestion detection based on redundant ACK invalid. But on the other hand, NDN also has its unique advantages when it comes to congestion control. First, there is an one-to-one correspondence between interest packet and data packet, so we can control and prevent the congestion at the receiving end by controlling the request rate. Second, intermediate nodes can also assist in congestion control, such as feed back network status information, control request forwarding rates and adjust forwarding strategies.

Existing works on NDN congestion control problem can be divided into three main types: receiver driven flow based control, hop-by-hop interest shaping control and the hybrid of both. Receiver driven flow based control performs in a TCP-like form and needs to maintain the state of each stream which brings a large load. It also does not work well when the content comes from multiple servers or multiple content caches. Hop-by-hop control adjusts the forwarding rate and forwarding path of interest packets at intermediate nodes, but it brings huge computational burden to the core networks and cannot resolve the root cause of the congestion, that is the interest requesting rate of users exceeds the capacity of network. Besides, some existing schemes do not pay attentions to content fairness and user fairness.

In order to effectively solve the congestion problem in NDN and avoid bringing large computational complexity to core networks at the same time, we propose our edge computing aided congestion control scheme which named EACC in this paper. The scheme can be divided into two main parts: (1) The detection and feedback of path congestion information. We detect the congestion at each intermediate node by monitoring the buffer occupancy, and mark the information in the data packets returned in real time for feedback. (2) Interest forwarding control at edge nodes. We formulate the forwarding control of the edge node as an MDP problem, and schedule interest packets based on the returned path congestion information and local user requests. The goal of each decision is to maximize the overall network revenue.

Compared with other existing solutions, we propose a new idea to solve the congestion control problem in NDN. The main contributions of this paper can be briefly summarized as follows:

- We propose an edge computing aided congestion control scheme (EACC) in this paper. The core computing unit is deployed at each edge router to control users' optimal interest forwarding based on local information and returned network congestion status. The forwarding process of edge node is modeled as an MDP problem, and its goal is to maximize the overall network revenue under the precise of avoiding congestion. We solve the decision problem using neuro-dynamic programming and present a practical implementation at edge nodes.
- We detect local congestion information at intermediate nodes by monitoring buffer occupancy and fill it into the new field of data packets to feedback the congestion status of corresponding path to edge routers. While ensuring the accuracy and timeliness of congestion detection, it also avoids bringing excessive computational complexity to intermediate nodes.
- We implement the proposed congestion control scheme in ndnSIM simulator and compare it with other two solutions, ICP and Best Route, which are receiver driven flow based control and built-in forwarding control respectively. Simulation results verify the effectiveness and advantages of EACC.

The rest of this paper is organized as follows: The background and related work are briefly described in Section II. Our proposed congestion control scheme is presented in Section III, followed by the performance evaluation in Section IV. Finally, we conclude the paper in Section V.

II. Background and Related work

A. Congestion Control in NDN

In NDN, each forwarding interface of router has a corresponding buffer. When data packets overflow the transmission buffer, congestion may occur in the network. That is, when the arrival rate of the data packet exceeds the processing rate of the router or the forwarding rate of the link, it will cause losses of data packets or interest packets due to congestion, and then trigger retransmission.

In order to avoid interest flooding, the size of data packet (or what we called data chunk) is usually set larger. Because of this, the congestion in NDN mainly comes from data chunks sent by content servers or the intermediate nodes where contents copies are cached rather than interest packets forwarded by users or downstream nodes. Data packets always return along the inverse path of the interest packets and there is an one-to-one correspondence between them, so we can schedule and control the returned data by shaping the forwarding of the interest packets.

B. Related Work

As mentioned before, existing congestion control works in NDN is mainly categorized into three groups: receiver driven flow based control, hop-by-hop control and hybrid control.

Receiver driven flow based control usually detects link congestion through timeouts or feedback signals. ICP [3] is a typical window-based receiver driven control scheme. It follows the idea of TCP congestion control and applies the AIMD (Additive Increase Multiplicative Decrease) strategy to window adjustment. ICP uses delay as a signal to detect congestion, and retransmits interests after timeout to ensure transmission reliability. Mahdian et al. presented MIRCC [4] which is a rate-based multi-path aware ICN congestion control approach. MIRCC proposes an algorithm to calculate the rate of each ICN link and and then introduces a method to make the scheme multi-path aware. As data content in NDN may come from multiple sources, there is no accurate RTT estimation, which makes the scalability of receiver driven flow based control is poor. In addition, receiver driven control is not sensitive to the occurrence of congestion and usually has a large lag.

Hop-by-hop control adjusts node's interest forwarding rate or policy based on the local congestion information detected at each intermediate node. HoBHIS [5] is a rate-based hop-by-hop congestion control mechanism, which calculates the available capacity of each CCN router in a distributed manner to adjust its session interest rate and therefore dynamically adjusts its data rate and transmission buffer occupancy. Wang et al. [6] analyzed the relationship between interest packets and data packets and pointed out that both the two kinds of packets have an impact on congestion. And based on this, they proposed a scheduling method that guarantees fairness. Hop-by-hop control is more like a best-effort forwarding adjustment which cannot fundamentally resolve congestion, because the root cause of congestion is that users' request rate is too high.

Hybrid control is usually a combination of the above two control ideas. Schneider et al. proposed PCON [7] which uses packet delay detection based on the CODEL queue without the need to assume link BDP and availability information are already known, and the user request rate adjustment and multi-path forwarding adjustment are both triggered by marked packets after congestion occurs. Yang et al. [8] added a penalty factor term to each congested link and modeled it as a global optimization problem according to the corresponding cost. The problem is solved using partial dual decomposition. Compared with the other two kinds, hybrid control is more comprehensive, but at the same time, it has higher complexity and brings additional communication overhead.

III. Our Proposed Scheme

In this section, we describe the design and implementation of our proposed EACC congestion control scheme.

A. Motivation and overview

Congestion control is a fundamental and very important problem in NDN, but it has not been well solved yet. While existing works are faced with more or less bothers of high computational complexity, multi-content source and multi-transmission path problems, we try to propose a solution to avoid these problems as much as possible while ensuring efficiency. The main design goals of our scheme can be summarized into the four points: lightweight, low latency, high efficiency and user fairness.

In order to achieve the above goals, we design EACC from two aspects. Firstly, we detect congestion by monitoring the buffer occupancy at intermediate nodes and the information is carried in the return data packets. This ensures EACC is lightweight and low latency. Secondly, we deploy the core computing unit at each edge router to control users' optimal interest forwarding. The forwarding process of edge nodes is modeled as an MDP problem and we give a practical solution using neuro-dynamic programming. Each user request will bring an independent event to trigger the decision, achieving the fairness between requests. These ensure the high efficiency and user fairness of EACC.

B. Detection and feedback of path congestion

Content Name	Signature&Signed Info	Data	Congestion tag	Congestion info

Fig. 1. Data Packet format

We first record and feedback the path congestion status information corresponding to each interface of edge nodes. The buffer occupancy is detected at each intermediate node, and when the local congestion occurs, the signal is transmitted to downstream by marking data packets. We add congestion-tag and congestion-info fields in the packet header to record the congestion information on the path, which is shown in Fig. 1. The tag field equals to 0 or 1 where 1 means congestion on at least one link on the entire path. And the info field is a 8-bit field, whose the i-th bit equals to 1 indicates that there is congestion on the i-th hop link from the current node. The processing steps of the info field are: First, when the node receives a data packet, if its congestion-tag equals to 1 and there are more than one egress faces, the info field will be shifted to the right by 1 bit. Second, if there is local congestion, the first bit of the congestion-info field of outgoing data packets will be set to 1 and turn the congestion-tag into 1. The overall architecture of the congestion information feedback is shown in Fig. 2. Edge router R1 will receive data packets with congestion information 01000000 from path R1-R2, and data packets with congestion information 10000000 and 10100000 from path R1-R5.

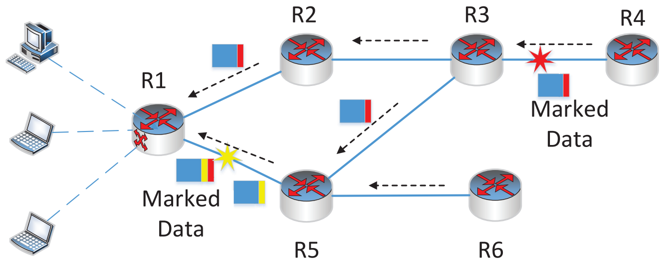

Fig. 2. Congestion info piggyback

When congestion occurs at a hop on the path, the specific information will be marked in data packets and delivered downstream until the edge node. When edge node receives these tagged packets, it calculates the total congestion cost on the egress path based on the information it carries.Then, the total path congestion cost at the edge node is calculated as:

$$cp = \sum_i \beta^{i-1} x_i, \tag{1}$$

where x_i represents the value of the i-th bit of the info field and β is the discount factor that is set to 0.7 in our scheme. Edge node uses the maximum congestion cost received in the average VRTT (Virtual Round-Trip Time) period as the final congestion cost of the egress path corresponding to the interface. When performing interest forwarding, we consider the actual available bandwidth of the current link as

$$b_l^{'} = \frac{b_l}{1 + cp_l}. \tag{2}$$

C. Interest forwarding control at edge nodes

1) Problem formulation: Based on the feedback congestion information and local user information, we formulate the interest forwarding control at edge nodes as an MDP (Markov Decision Process) problem similarly to [9]. For a given edge router, it is assumed that the router has a set of interfaces denoted by $L = \{1, 2, \ldots, l, \ldots, L\}$, where the egress path bandwidth corresponding to the interface l is B_l. For the contents in NDN, we assume that they can be divided into $M = \{1, 2, \ldots, m, \ldots, M\}$ categories in total, and for the same type of content, they have the same bandwidth requirements and benefits. For a request for category m content, suppose its bandwidth requirement is b_m, and the problem is modeled as follows.

Let $s_{lm}(t)$ denotes the number of m-type requests that have been forwarded but have not received data on path l at time t, then the state space S is:

$$\sum_m s_{lm}(t) * b_m \leq B_l, l \in L. \tag{3}$$

Then we define three basic events and two additional events, that is, the correction events caused by the congestion cost are: the basic events include $e_{l,m,0}$, $e_{l,m,1}$ and $e_{l,m,2}$. $e_{l,m,0}$ indicates that the corresponding data from the upstream is received, $e_{l,m,1}$ indicates that the request can be matched locally, otherwise it is indicated as an event $e_{l,m,2}$ which means select the path l to forward the user request. The correction events include $e_{l,0,1}$ and $e_{l,0,0}$. $e_{l,0,1}$ indicates that the congestion cost increases, which is equivalent to a virtual request on the path l without return of data, and $e_{l,0,0}$ indicates that the congestion cost decreases, which is equivalent to that some or all of the virtual requests are satisfied. The whole event sets are represented as:

$$E = \{e_{l,m,0}, e_{l,m,1}, e_{l,m,2}, e_{l,0,0}, e_{l,0,1} | \forall l \in L, \forall m \in M\}.$$

Based on the above types of events, the corresponding set of actions is as follows

$$A = \{a_0, \dots, a_l, \dots, a_{L+l}, \dots, a_{2L+1}, a_{2L+2}\},$$

which respectively represents the actions "response to content requirements", "choose path l to forward requests", "choose path l to forward virtual requests", "response to virtual requests" and "reject requests". To be noticed, event $e_{l,m,0}$ or event $e_{l,0,0}$ will not trigger a decision but just update the network status of the node.

The reward function $r(\boldsymbol{s}, e, a, \boldsymbol{s}')$ is defined as bellow

$$r(\boldsymbol{s}, e, a, \boldsymbol{s}') = I_1 * (c_m + u_{l,m}) + I_2 * v_{l,m} - I_3 * p_m, \quad (4)$$

where p_m is the rejection cost when the interest is denied by edge router, c_m is the transmission benefit which equals to the content provider's economy benefit, $u_{l,m}$ represents the user benefit when the request is forwarded through path l. $v_{l,m}$ equals to 0 when l is the corresponding feedback path, otherwise it is negative. I_1, I_2, I_3 are three 0 or 1 variables which satisfy $I_1 + I_2 + I_3 = 1$. When they are equal to 1, they respectively stand for forwarding ordinary interest , forwarding virtual interest and rejecting interest.

After defining the reward function $r(\boldsymbol{s}, e, a, \boldsymbol{s}')$, the change in state $\boldsymbol{s}(t)$ is a finite-state continuous-time MDP under a given policy μ, and $\boldsymbol{s}(t_0)$ is the initial state. The long-term average reward of the process with strategy μ is

$$v(\mu, \boldsymbol{s}_0) = \lim_{N \to \infty} \frac{1}{t_N} * E(\sum_{n=0}^{N} r(\boldsymbol{s}(t_n), e(t_n), a(t_n), \boldsymbol{s}(t_{n+1})).$$

The most direct solution is the reverse dynamic programming algorithm, which needs to traverse all feasible actions in each state. If for any policy x, there is $v(\mu^*) \geqslant v(\mu)$, the policy μ^* is called the average cost optimal policy, and the optimal performance value is η^*. The corresponding Bellman [10] equation is

$$v^* \tau(\boldsymbol{s}) + h^*(\boldsymbol{s}) = E(\max_{a \in A}[r(\boldsymbol{s}, e, a, \boldsymbol{s}') + h^*(\boldsymbol{s}')]), \quad (5)$$

$h^*(\boldsymbol{s})$ is the optimal state value function, and $\tau(\boldsymbol{s})$ is the state dwell time. The optimal policy μ^* can be obtained as

$$\mu^*(\boldsymbol{s}, e) = arg \max_{a \in A}[r(\boldsymbol{s}, e, a, \boldsymbol{s}') + h^*(\boldsymbol{s}')]. \quad (6)$$

2) Neuro-Dynamic Programming solution: Standard dynamic programming can be used to solve the above model, but it requires complex calculation and may cause dimensional disaster. So here we consider using neuro-dynamic programming (NDP) [11] [12] to solve this MDP model. The architecture of NDP is shown in Fig. 3, which includes four parts: feature extraction, function approximation, parameter update, and decision module.

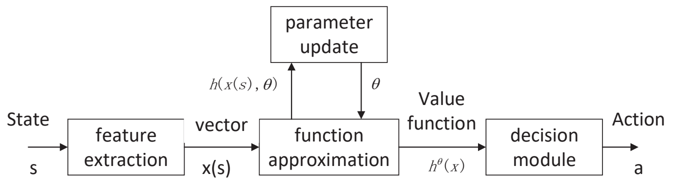

Fig. 3. Architecture of NDP

Feature extraction refers to the process of transforming the input state $\boldsymbol{s}$ into a feature vector $\boldsymbol{x}(\boldsymbol{s})$. This module is the core module of the NDP algorithm and the goal of feature extraction is to represent the feature vector with the smallest dimension as possible without feature loss. Based on the characteristics of NDN, we here extract the feature vector as follows. x_{lm} represents the bandwidth consumption on path l caused by m type contents and $x_{lm} = s_{lm} * b_m$. x_l represents the total bandwidth consumption on path l and $x_l = \sum_m x_{lm}$.

$$\boldsymbol{x}(\boldsymbol{s}) = [1, \dots, 1, x_{11}, \dots, x_{lm}, \dots, x_{LM}, x_1^2, \dots, x_L^2]^T$$

The calculation complexity is more considered when selecting approximation function, so we choose a linear function as the approximate state value function here with $\widetilde{h}(\boldsymbol{s}, \boldsymbol{\theta}) = \boldsymbol{\theta x}(\boldsymbol{s})$. $\boldsymbol{\theta}$ is a parameter vector defined as

$$\begin{aligned}\boldsymbol{\theta} = [\theta(1), \dots, \theta(L), \theta(11, b), \dots, \theta(lm, b), \dots, \theta(LM, b),\\ \theta(1, b, b), \dots, \theta(L, b, b)].\end{aligned}$$

Then the approximate function can be expressed as

$$\widetilde{h}(\boldsymbol{s}, \boldsymbol{\theta}) = \sum_l \theta(l) + \sum_l \sum_m [\theta(lm, b) x_{lm}] + \sum_l [\theta(l, b, b) x_l^2].$$

The main task of the parameter update module is to reduce the error between the approximate value function and the optimal value function. Parameter update is a supervised iterative process with feedback and here we use the simplest $TD(0)$ method to update parameter $\boldsymbol{\theta}$. The k-th update rule and the optimal average reward $\widetilde{v}$ are defined as follows

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \gamma_k d_k \bigtriangledown_{\boldsymbol{\theta}} \widetilde{h}(\boldsymbol{s}_{t_{k-1}}, \boldsymbol{\theta}_{k-1}), \quad (7)$$

$$\widetilde{v}_k = \widetilde{v}_{k-1} + \eta_k (r(\boldsymbol{s}_{t_{k-1}}, e, a_{t_{k-1}}, \boldsymbol{s}_{t_k}) - (t_k - t_{k-1}) \widetilde{v}_{k-1}),$$

where γ_k and η_k are the small step size parameters, d_k is the time difference, and $\bigtriangledown_\alpha$ can help accelerate the convergence. The definition of d_k is

$$\begin{aligned} d_k = r(\boldsymbol{s}_{t_{k-1}}, e, a_{t_{k-1}}, \boldsymbol{s}_{t_k}) - (t_k - t_{k-1}) \widetilde{v}_{k-1} \\ + \widetilde{h}(\boldsymbol{s}_{t_k}, \boldsymbol{\theta}_{k-1}) - \widetilde{h}(\boldsymbol{s}_{t_{k-1}}, \boldsymbol{\theta}_{k-1}). \end{aligned} \quad (8)$$

Substituting the optimal difference function $\widetilde{h}(\boldsymbol{s}, \boldsymbol{\theta})$ above to obtain the optimal strategy is

$$\mu^{\boldsymbol{\theta}}(\boldsymbol{s}, e) = arg \max_{a \in A}[r(\boldsymbol{s}, e, a, \boldsymbol{s}') + \widetilde{h}(\boldsymbol{s}', \boldsymbol{\theta})]. \quad (9)$$

3) Decision process: Finally, the decision process can be summarized in the following **Algorithm 1**:

Algorithm 1: Decision process of interest forwarding control at edge nodes.

```
   Input: initial state: $\boldsymbol{s}_0$, initial parameter vector: $\boldsymbol{\theta}_0$;
   initial approximate value function: $\widetilde{h}(\boldsymbol{s}_0, \boldsymbol{\theta}_0)$;
   set step length sequence $\gamma_i = 1/i, \eta_i = 1/i$;
   Iteration number $i = 1$;
1  while given $\boldsymbol{s} \in \boldsymbol{S}, e \in E$ do
2      //new interest arrival event
3      if $e = e_{l,m,2}, l \in L, m \in M$ then
4          obtain an approximate optimal solution $\mu^{\theta}(\boldsymbol{s}, e)$
           according to equation (9),
5      end
6      //congestion price increase event
7      if $e = e_{l,0,1}, l \in L$ then
8          turn the event into series of virtual requests on the
           corresponding path, obtain an approximate optimal
           solution $\mu^{\theta}(\boldsymbol{s}, e)$ according to equation (9),
9      end
10     //congestion price decrease or data event
11     if $e = e_{l,0,0}$ or $e_{l,m,0}, l \in L, m \in M$ then
12         update network status and free up bandwidth,
13     end
14     Update transfer status after taking action and recalculate
       the parameter vector $\boldsymbol{\theta}$ according to equation (7), and then
       modify the approximation value function $\widetilde{h}(\boldsymbol{s}, \boldsymbol{\theta})$.
15     Increase the iteration number $i = i + 1$.
16 end
```

D. User sending rate adjustment

For the user's own sending rate, we use the modified traditional AIMD sending window adjustment strategy. The growth of the user window is triggered by each return of data, and the decrease is triggered by feedback information or RTO timeout. When the resource reaches the upper limit, edge nodes will reject new requests and then feedback signals to users. And because the cost and latency of interacting with edge nodes is low, it can speed up in the window growth stage to increase the utilization of edge access links. Besides, it is shown in subsection-C that the scheduling of edge nodes is triggered by single request event, so the scheme can ensure that requests of different users are fair and the adaptability of user access and departure.

IV. Performance Evaluation

In this section, we evaluate the performance of our proposed scheme. We conduct experiments based on ndnSIM (version 2.7, a ns-3 based NDN simulator) under the Ubuntu 18.04 virtual machine and compare our scheme to ICP [3] and BestRoute. ICP is a typical receiver driven control method and BestRoute is the built-in forwarding scheme in NDN. The simulation topology is shown in Fig. 4, where "ER" represents an edge router, "R" represents a core router, and "CP" corresponds to a content provider. The topology includes the several typical scenarios in NDN such as multi-source, multi-path and shared bottleneck links, so it can also be extended to more complex topologies.

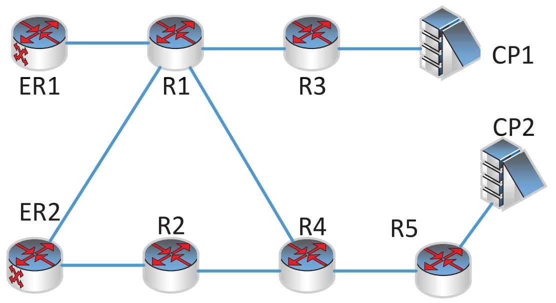

Fig. 4. Simulation scenario

Here we set the capacity of each link to 10Mbps and the propagation delay to 10ms. Each edge node deploys two users to send interest packets at the same rate, and the sending rate is equal to the number of requests per user per second. There are four types of content, and the data size corresponding to each content is 1, 2, 3, and 4 KB respectively. The user's requests for different contents are distributed in proportion. We set up different initial user interest sending rates and conduct serval simulations. Each simulation lasts for 10s. Then we compare the different schemes in terms of content acquisition delay, end-to-end retransmission and link utilization.

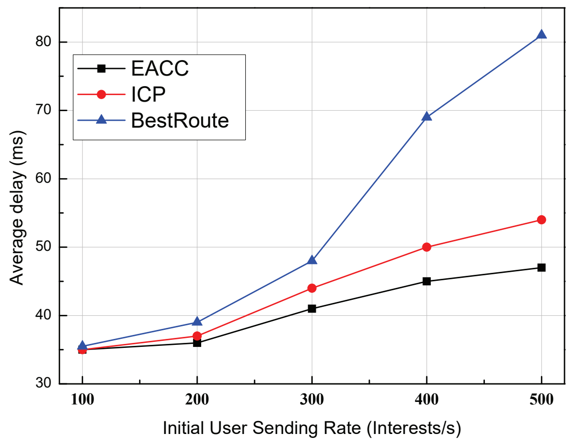

Fig. 5. Average data acquisition delay

We first compare the content acquisition delay with different initial user sending rates. The average end-to-end delay refers to the average of data acquisition time of all the interest requests sent by four different consumers accessed to the edge routers respectively. As it is shown in Fig. 5, EACC has the best performance, and this advantage is more obvious when the user sending rate is higher. This is because EACC has effectively controlled the forwarding of interest in the vicinity of the source, and only continues to forward requests when the link resources are sufficient, ensuring that each sent request can be effectively satisfied. ICP can achieve good results when the network resources are sufficient, but it can not deal well with complex multi-path scenario when the traffic load is heavy. BestRoute only regulates and controls the forwarding of interest packets at intermediate nodes, which is a best-effort solution. It cannot effectively solve the problem from the root cause that is user requests exceed the network capacity.

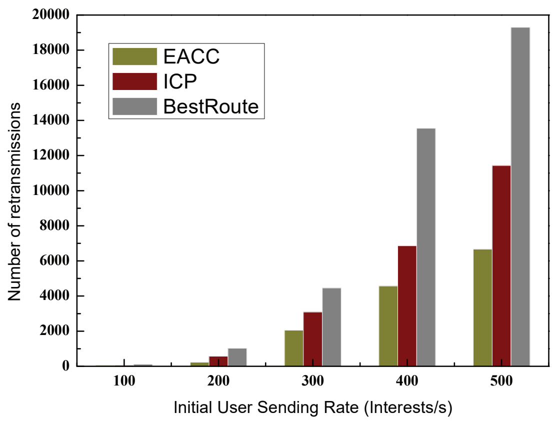

Fig. 6. Number of end-to-end retransmissions.

Secondly, we compare the number of end-to-end data retransmissions, that is the number of users' re-requests caused by packet losses or timeouts. As we can see from Fig. 6, when network resources are sufficient, all three schemes have good performance. But with the increase of user sending rate, the performances of ICP and BestRoute decrease significantly. This is because they, especially the BestRoute strategy, cannot effectively control the number of user requests injected into the network.The returned data packets will compete with each other for link resources, resulting in a large number of packet losses and retransmissions. On the contrary, EACC filters user requests at edge nodes, effectively reducing transmission pressure and avoiding data transmission exceeding network capacity.

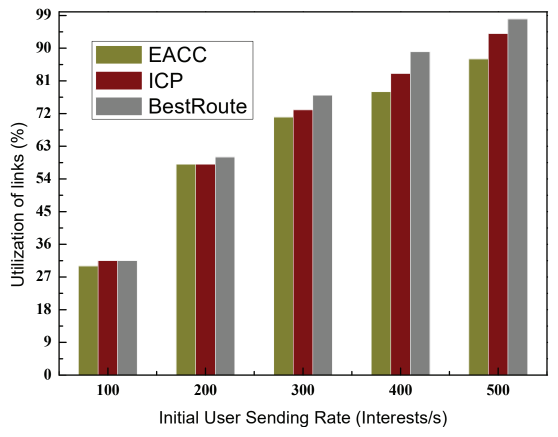

Fig. 7. Average utilization ratio of links

Finally, we compare the link utilization ratio under the three different schemes. As shown in Fig. 7, BestRoute has the highest average links utilization. As users' request rates increase, the traffic load of links under BestRoute strategy will increase rapidly and cause packet loss and large content acquisition delay. ICP adjusts the user sending rate according to network conditions, which can also reduce the traffic load. As for EACC, the main idea is to avoid congestion. EACC controls the traffic load in the whole network by filtering interests at edge nodes, thereby ensuring lower transmission delay and quality of service.

V. Conclusion

In this paper, we proposed a novel scheme based on path congestion detection and edge forwarding control to solve the congestion control problem in NDN. We performed congestion detection at each intermediate node and feeded the information back to edge nodes hop by hop through the marked data packet. We formulated the interest forwarding control at edge node into a MDP problem based on the local request information and feedback congestion status, and we solved the problem with neuro-dynamic programming algorithm using the TD(0) update method. Finally, we evaluated the performance of our proposed mechanism by comparing it with two other methods. Simulation results show that our scheme has good performance in terms of delay, data retransmission and link utilization.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371 and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. 2016394.

References

[1] L. Zhang, A. Afanasyev, J. Burke *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[2] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, Aug. 1988.

[3] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp: Design and evaluation of an interest control protocol for content-centric networking," in *2012 Proceedings IEEE INFOCOM Workshops*, March 2012, pp. 304–309.

[4] M. Mahdian, S. Arianfar, J. Gibson, and D. Oran, "Mircc: Multipath-aware icn rate-based congestion control," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, New York, NY, USA, 2016, p. 110.

[5] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *2012 Proceedings IEEE INFOCOM Workshops*, March 2012, pp. 322–327.

[6] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 55–60, Aug. 2013.

[7] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for named data networking," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, p. 2130.

[8] W. Yang, Y. Qin, and Y. Yang, "An interest shaping mechanism in ndn: Joint congestion control and traffic management," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[9] J. Yao, B. Yin, and X. Tan, "A smdp-based forwarding scheme in named data networking," *Neurocomputing*, vol. 306, pp. 213 – 225, 2018.

[10] R. Bellman, "On the theory of dynamic programming," *Proc Natl Acad Sci*, vol. 38, no. 8, pp. 716–719, Aug. 1952.

[11] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis, "Call admission control and routing in integrated services networks using neuro-dynamic programming," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 2, pp. 197–208, 2000.

[12] K. Zhu, Y. Ran, E. Yang, and J. Yang, "Joint admission control and routing via neuro-dynamic programming for streaming video over sdn," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 20–25.