PLR: An In-Network Proactive Loss Recovery Scheme for Named Data Networking

Yuxin Chen*, Jiayu Yang*, Xuanbo Huang*, Jiangping Han*, Bobo Wang*†, Jian Li*, Kaiping Xue*
*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China
†Corresponding author, B. Wang (wangbobo@ustc.edu.cn)

2023 32nd International Conference on Computer Communications and Networks (ICCCN) | 979-8-3503-3618-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCCN58024.2023.10230118

***Abstract*—With potential advantages over TCP/IP for content delivery, mobility, and security, Named Data Networking (NDN) has become a promising architecture for the next-generation network. However, its poor performance in reliable transmission is still an unsolved problem. Many existing schemes in NDN employ inaccurate retransmission timeouts calculated with RTTs from diverse content sources to detect packet loss, which is lagging and may deteriorate transmission performance. Besides, after identifying the loss, the consumer costly resends the request to recover it, further increasing recovery time. In this paper, we propose an in-network Proactive Loss Recovery (PLR) scheme, which provides an efficient in-network method for timely detection and proactive recovery of lost packets. Deployed on each router, PLR detects the loss by monitoring queue status and sends high-priority explicit feedback to notify consumers of loss events timely. Meanwhile, lost packets are stored in each router's cache and will be retransmitted at an adaptive rate based on the detected remaining bandwidth. The simulation shows that PLR can vastly reduce the number of retransmissions on consumers, and the content completion time can be decreased by up to 21.8% compared with the baseline.**

***Index Terms*—Named Data Networking, reliable transmission, loss recovery**

I. Introduction

With the development of the Internet, its dominant service has shifted to content delivery and requires a more flexible approach to content distribution and mobility support. According to Ericsson Mobility Report [1], the number of mobile devices grows rapidly, and video traffic will account for 79% of all mobile data traffic by 2027. To avoid redundant transmission and provide better mobility support, Information-Centric Networking (ICN) [2] brings a new network architecture that decouples the content and its network location. The architecture allows each content to be individually cached, requested, transmitted, and protected anywhere in the network. Named Data Networking (NDN) proposed by Zhang *et al.* [3] is the most promising implementation of the ICN architecture. It turns the connection-oriented transport of TCP/IP architecture into receiver-driven connectionless transport. Additionally, with the in-network cache and stateful data plane, consumers can obtain data more flexibly from multiple content sources by sending Interest requests.

Although NDN is a brand new network architecture that enables dynamic utilization of network resources, it also brings some problems. Among them, its poor performance in reliable transmission is an unsolved one. Usually, fast detection and timely recovery of lost packets are required to improve transmission reliability. There are already mature and practical schemes in the TCP/IP architecture [4]. For example, TCP senders can trigger immediate retransmissions on receiving three duplicate acknowledgments (ACK) to fast recover the loss. They can also maintain a Retransmission Timeout (RTO) calculated from Round-Trip Time (RTT) to handle the retransmission failure when the network is congested. Unfortunately, for having an entirely different transport mode and characteristics from TCP/IP, those schemes are either invalid or ineffective in the NDN network [5]. The specific reasons are as follows. First, NDN adopts a receiver-driven transport mode, in which the consumer sends an Interest packet to obtain a Data packet, and there is no ACK to notify the data transmission status. Therefore, the retransmission based on ACK is disabled in NDN. Second, since Data packets may come from multiple content sources, consumers only get unstable RTTs, which makes it difficult to determine an accurate RTO. Therefore, the RTO-based retransmission works but is usually inefficient in NDN. Furthermore, to recover a lost packet, the consumer must first wait for the RTO to determine the loss and then resend the Interest packet, which has to go through a complete data retrieval process, resulting in a pretty long recovery time. Overall, directly utilizing schemes of TCP/IP architecture causes inaccurate loss detection and longer recovery time. They may exacerbate link congestion for those loss-based congestion control algorithms and result in massive packet loss, further increasing the completion time of content.

There are some schemes have been proposed to improve the transmission reliability of the NDN network. We divided them into two categories based on the type of packet loss they addressed. The first category is the hop-by-hop reliability schemes [6]–[8], which mainly leverage link-layer mechanisms to detect incomplete packets and recover lost segments from the previous hop. They are effective and have low random loss rates in wireless scenarios. However, when the loss is caused by congestion and burst, these schemes are powerless as recovery packets are overwhelmed by excessively transmitted packets. The second category is helpful for lost packets caused by congestion, mainly congestion control algorithms [9]–[14]. They strive to output more reasonable transmission rates to reduce congestion and packet loss. Unfortunately, the ideal control is impossible to achieve, as a number of consumers have uncontrollable and unpredictable actions to start or cease transmission. Faced with network congestion and burst transmission, they can only utilize the unreliable and costly timeout retransmission mechanism to detect and recover lost packets with poor performance. Therefore, we urgently need an efficient loss recovery scheme customized for NDN to timely detect and recover the congestion-induced loss, further decreasing content completion time.

In this paper, we design an in-network Proactive Loss Recovery (PLR) scheme to solve the reliable transmission problem in the NDN network. PLR offloads the detection and recovery for congestion-induced loss to routers, which is feasible in NDN routers with stateful forwarding planes. Specifically, PLR leverages each router to provide timely and accurate detection of congestion-induced loss by monitoring its queue. Then, packets out of the queue are stored in each router's cache before being dropped. Meanwhile, immediate notifications are delivered to consumers for congestion. Thus, consumers can adjust the transmission rate in time to avoid further performance deterioration. Afterward, when observing that the congestion has eased, the router actively retransmits the stored lost packets at an adaptive rate. The main contributions are summarized as follows:

- We propose an in-network reliable transmission scheme, called PLR, customized for NDN. PLR is a module deployed on each router that can detect congestion-induced loss, timely notify consumers, and recover the loss without consumer participation. Moreover, PLR measures the remaining bandwidth and adjusts the retransmission rate during the recovery accordingly, thus avoiding congestion exacerbation.
- We theoretically analyze the behaviors of PLR and the conventional loss recovery scheme in terms of their detection accuracy and recovery time. PLR can avoid the problem of RTT interference from multiple sources in RTO-based loss detection. Moreover, PLR can reduce recovery time by RTO compared with the conventional scheme in most cases by cutting the loss recovery loop.
- We implement PLR in the ndnSIM with new extended data structures and processing methods. The simulation results show that PLR can significantly reduce the number of Interest retransmissions and decrease the completion time by up to 21.8% compared with the baseline.

The rest of this paper is organized as follows. Section II introduces the background and motivation of this paper. The related works and their analysis are present in Section III. We describe the design and implementation of our scheme in Section IV and give a more profound analysis in Section V. Subsequently, we give performance evaluation in Section VI. Finally, we conclude our work in Section VII.

II. Background and Motivation

This section includes an overview of the NDN network and the challenges when it suffers packet loss. Then, we present our motivation.

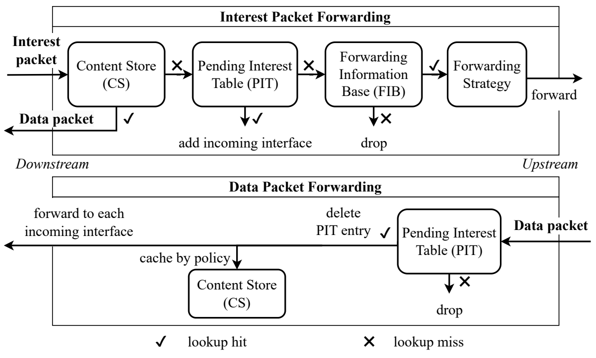


Fig. 1: Forwarding process at an NDN node.

A. NDN Network

In NDN, there are two basic types of packets: Interest packet and Data packet. An Interest packet contains the requested content's name, and a Data packet contains both the name and the content. Generally, an NDN router maintains a forwarder and three data structures: a Content Store (CS), a Pending Interest Table (PIT), and a Forwarding Information Base (FIB). The specific forwarding process at an NDN router is as follows:

As shown in Fig. 1, when an Interest packet arrives, an NDN router first uses the name in the Interest packet to check the CS for matching data. If the search for Data packets in CS hits, the router returns the matched Data packet on the interface from which the Interest came. Otherwise, the router looks up the name in its PIT, which records the interfaces that a pending Interest packet comes from and goes to. If a matching entry exists, the router adds the new incoming interface to the PIT entry. Otherwise, the router performs the longest prefix match in the FIB to search available egresses. Then, based on the forwarding strategy, the router forwards the Interest toward the content sources by selecting one or more matching egresses. When a Data packet arrives, the router searches the PIT for all the interfaces from which the corresponding Interest packet comes and sends the Data packet to each of them. Then, the router deletes the PIT entry and decides whether to cache the Data packet in its CS based on cache policy. The stored content may accelerate follow-up data delivery. In the standard transmission, NDN provides a receiver-driven "pull-based" data request mode and a one-Interest-one-Data delivery principle. Thus, the response source of the data packet is no longer fixed.

B. Loss Recovery Challenges in NDN

NDN brings many new features, thus providing more flexible content acquisition. However, it also brings some problems, and one of the main problems is its inefficient loss detection and costly loss recovery for still using RTO to detect and recover the loss [11], [15]. Next, we illustrate the problem by presenting the specific packet loss detection and recovery process with a simple example.

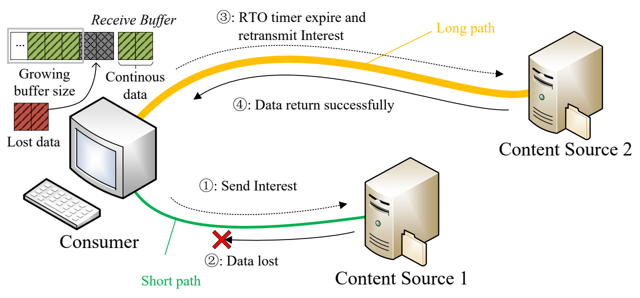


Fig. 2: The loss recovery process in NDN.

As shown in Fig. 2, there are two content sources that serve the same content through two paths with different delays. The consumer sends Interest packets to request the content. Then, the router selects the path for each Interest packet adaptively according to its forwarding strategies [16], [17], enabling the consumer to concurrently obtain content from two sources. Supposing the consumer sends an Interest packet and is forwarded to *Content Source 1* via the short path, but the replied Data packet is lost on the way back. After a period of time up to RTO, the consumer knows that the packet is lost during transmission, and then it resends the Interest packet to recover the packet. In this process, RTO is the core of loss detection, which determines the accuracy and timeliness of the detection. Generally, the consumer calculates RTO using the following equation ported from TCP [18]:

$$rto = srtt + k \cdot rttvar,$$

where $srtt$ and $rttvar$ are the smoothed value and the estimated variance of RTT, respectively, and k is a magnification factor. However, in the NDN network, the result of the calculation of RTO is usually too large because the mixed RTT value from two paths has a larger variance. Moreover, the retransmitted Interest packet has to be forwarded upstream again through a re-chosen path, which results in a long and unstable recovery time. Worse still, many practical applications require continuous series of data, so consumers have to buffer out-of-order packets. Consequently, longer loss detection and recovery time lead to larger buffer requirements. If the consumer has limited buffer space, those unrecovered lost packets may block the transmission and extend the completion time.

To summarize, in current strategy, consumers have to wait for the RTO timer to detect the loss and go through a complete content retrieval procedure to recover the lost packet. It is costly and may deteriorate transmission performance.

C. Motivation

We summarize the two main aspects that have a great impact on loss recovery in NDN:

- Passive loss detection using RTO by the consumer is inaccurate and lagged.
- Loss recovery by retransmitting the Interest packet is costly and time-consuming.

The inaccurate loss detection can lead to misjudgment of congestion and make inappropriate rate adjustments. The costly recovery method requires the network to forward packets repeatedly, which may further exacerbate congestion and deteriorate transmission performance. The problem is more pronounced when the loss is caused by congestion. Therefore, our PLR scheme focuses on the detection and recovery of the loss due to congestion and provides a more efficient method.

According to our analysis, we observe that the root cause of both problems is that the detection and recovery functionalities are only consumer-based, which is inevitably passive and costly. Therefore, PLR fully utilizes the in-network caching and stateful forwarding plane of NDN network to offload loss detection and recovery to intermediate routers.

Before providing a consumer-imperceptible loss recovery scheme, we first analyze how congestion-induced loss occurs. Same as TCP/IP network, NDN is also built on packet switching architecture, where packets are queued in routers' buffer to be forwarded one by one. When packets to be forwarded are generated faster than the forwarding capacity of routers, their buffer will continue to grow and finally overflow. Packets that exceed the queue are discarded by the router according to the queue management policy. Therefore, PLR is deployed on each router and cooperates with the queue management policy to detect the occurrence of loss due to congestion, thus ensuring accurate and timely loss detection. Then, PLR sends a notification packet to feed the loss event back to the consumer through the reverse path maintained in PIT.

In terms of loss recovery, PLR proactively recovers lost packets at the location where they are detected. To do so, PLR carves out part of the router's cache to store detected lost packets for later retransmission. Since the retransmission during congestion is likely to fail and aggravate congestion, PLR provides a congestion-aware trigger mechanism. Besides, it estimates the remaining bandwidth and provides an adaptive retransmission adjustment to ensure successful retransmissions while avoiding interruptions to normal transmission.

III. Related Work

In this section, we review some related works that try to overcome the problem caused by packet loss. In general, the related works can be divided into two categories based on the type of packet loss they addressed.

The first category is represented by hop-by-hop reliability mechanisms, which detect and recover lost packets hop-by-hop and are dedicated to random packet loss. Vusirikala *et al.* [6] proposed BELRP, which operates at each interface of a router. It assigns a sequence number to every frame when sending and expects ACK from the next hop. It uses gaps in the acknowledged frame sequence to confirm the loss and attempts to retransmit the lost frame immediately. If the loss is continuously detected, the router repeats retransmission attempt several times before giving up. STNDN [7] improves the efficiency of BELRP by dynamically selecting the deployed nodes. WLDR [19] provides a similar mechanism to detect the loss but uses explicit feedback to trigger retransmission. L4C2

[8] also adopts the same detection method and uses cache and network coding to recover loss hop-by-hop. These schemes work well in recovering random packet loss. However, they play a limited role in congestion-induced loss recovery because they all take immediate retransmission to recover detected loss before congestion mitigates. Considering the inevitable gap between the occurrence and mitigation of congestion due to the control loop, the multiple retransmission attempts hardly take effect. Moreover, the retransmission during congestion will compete with the currently transmitted packets for bandwidth, which will further aggravate congestion and deteriorate performance.

The second category is congestion control algorithms, which play an essential role in reducing congestion-induced loss. ICP [9] is one of the earliest congestion control algorithms proposed for NDN. It increases the congestion window additively with each packet received and decreases it multiplicatively when RTO detects a loss. However, the RTO is determined by averaging the maximum and minimum RTT values from the last 20 packets on a single path. It does not consider the effect of cache, leading to a small RTO that can cause spurious timeouts and unnecessary window decrease [20]. The schemes maintaining a separate RTO for each path [21] or each content source [14] are also subject to a similar problem because they cannot predict the replied content source due to the unstable cache and dynamic forwarding. There are also schemes that utilize congestion signals instead of RTO-based loss events to solve the problem. PCON [10] is the most representative algorithm among them. It detects congestion by measuring the queue length in every router and marks Data packets in the way of CoDel, an advanced Active Queue Management (AQM) scheme, to notify downstream routers and consumers. The router transfers part of the traffic to other interfaces once receiving a marked Data packet, and the consumer also adjusts windows according to it. PCON avoids the effect of inaccurate RTO to some extent. In BBR-CD [12], the consumer follows the principle of BBR that adjusts transmission rate by probing the bandwidth-delay product. In 3CP [13], the consumer employs a per-packet feedback computation to inform consumers of the available resource of paths toward the content source. In IEACC [11], it divides mixed Data packets into different congestion degrees by a lightweight clustering algorithm and feeds congestion information into the deep reinforcement learning model to optimize the window adjustment. Undeniably, all these algorithms can reduce the loss caused by congestion. However, the network is constantly changing due to unpredictable user behaviors, which may cause possible abnormal control and burst transmission. Although well-designed, these schemes still face the packet loss problem. They still use a backup RTO-based loss detection and retransmit requests in the consumer to recover the loss, which is ineffective and costly.

To sum up, congestion control algorithms can reduce congestion-induced loss in most cases. However, they are still subject to loss when suffering abnormal control and burst transmission. Therefore, we propose PLR that provides a feasible and effective in-network method. PLR aims at proactively detecting and timely recovering the congestion-induced loss to complement existing congestion control schemes in reducing loss detection and recovery time, thus improving transmission performance. In the next section, we present design details.

IV. Proactive Loss Recovery

A. Overview

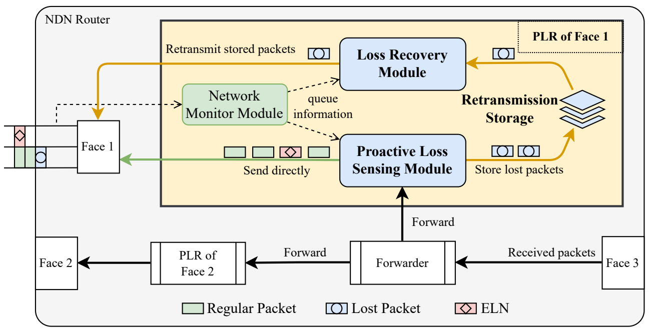


Fig. 3: The architecture of PLR.

Firstly, we present the design overview of PLR. Fig. 3 illuminates its architecture. PLR is deployed on NDN routers, and there is an independent PLR instance at each interface of each router to process all packets from the forwarder. We design a new type of packet, Explicit Loss Notification (ELN), and a new data structure Retransmission Storage (RS). Except for the auxiliary Network Monitor Module (NMM), PLR consists of two main controlling modules: the Proactive Loss Sensing Module (PLSM) and the Loss Recovery Module (LRM).

Specifically, NMM is a unified functional abstraction that provides access to the queue to get accurate and real-time information on the interface for the other two modules. In practice, the implementation of NMM is feasible but platform-specific, such as compiling a kernel module to read the queue information from the kernel network stack in soft routers. PLSM relies on NMM to acquire queue information to provide more accurate and timely loss detection and notification. It detects loss by checking if the queue is full. If true, it will bypass the regular transmission pipeline and store the packet in the RS. Meanwhile, PLSM constructs high-priority ELN packets carrying the names of stored packets to notify the consumer of loss events timely. The ELN will follow the same reverse path as the packet back to the consumer but preserve the path. LRM periodically acquires the queue information and interface statistics through NMM and forwarding core to monitor the congestion status and estimate the remaining bandwidth. When congestion mitigates, it recovers the lost packets stored in RS at an adaptive rate calculated from the estimated remaining bandwidth, which prevents overload and ensures successful retransmissions.

We will present detailed descriptions of the new packet type ELN, the new data structure RS, and the two main modules, PLSM and LRM, in the following sections, respectively.

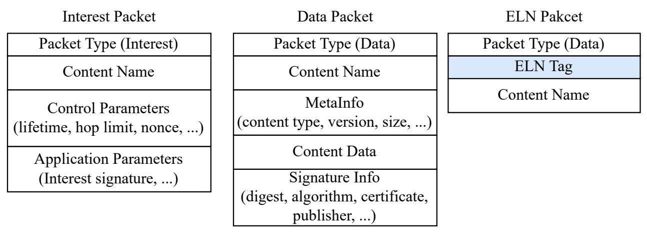


Fig. 4: The structure of Interest, Data and ELN packet.

B. Explicit Loss Notifications

To notify the consumer of the congestion-induced loss detected by the router, we design a new type of packet ELN. In Fig. 4, we show the structure of all packet types, and we can find the ELN is extremely concise compared to others, which contains only three minimal fields: packet type, ELN tag, and content name. The packet type of ELN is actually marked as “Data”, and the ELN tag field is used to distinguish the ELN from the ordinary Data packet. The content name of ELN is the same as that of lost packets it is to notify.

The same name as the lost packet and “Data” packet type allows an ELN to fully utilize the existing Data packet forwarding pipeline, which means it can return to the consumer along the existing reverse path without any extra operation. However, in order to recover the lost packets in-network, routers can not delete the corresponding PIT entry after forwarding the ELN like ordinary Data packets. So the ELN tag is used to skip the deletion of PIT entry and unnecessary caching operations in practical implementation.

In order to prevent the ELN from being affected by possible queuing on the path and reach consumers in time, the ELN has a higher priority than ordinary packets. We leverage the QoS mechanism to implement the feature. Specifically, we deploy a priority queue at each interface of the router, where the high-priority queue preempts the low-priority queue unconditionally. We further design a packet filtering system at the link layer to identify the encapsulated NDN packet type and assign high priority only to ELN. In this way, ELN packets will unconditionally preempt resources on each node for fast forwarding, ensuring their efficiency. As for the overhead, it is extremely low because its size is determined by the content name length, which is much smaller than the packet carrying the actual content, and its amount is limited by the trigger mechanism we’ll discuss below. So it requires only a few forwarding resources for efficient loss notification.

C. Retransmission Storage

RS is a data structure designed to store lost packets for in-network recovery. RS is created in the same location as the general cache, usually in memory, which can be regarded as a reserved part of the cache space. The maximum size of a RS is determined as follows:

$$len_R = \alpha \times len_Q, \tag{1}$$

where len_R and len_Q are the maximum size of RS and queue, respectively. The parameter α is a factor between 0 and 1 that controls the size of the RS, and we will discuss the effect of different values of it in Section VI-B. An item in the RS is composed of the actual packet and a timer, where the timer is used to limit the maximum residence time of the packet to avoid continuously occupying the storage space under harsh network conditions. There is no replacement policy for RS, so the insertion operation will fail when the RS is full. Moreover, when reading a packet from RS, it will return the packet with the shortest remaining time, known as the first-in-first-out feature. To some extent, RS is similar to an extension of the interface queue, but they are not the same. One reason is that RS is created at the network layer, where the storage is different from the queue. Another is that all packets in the queue are eventually sent, while packets in RS are controlled by the loss recovery module of PLR and may not be sent due to the timer mechanism. In our design, there may be multiple RS on a router because we create an individual RS for each instance of PLR. So the total space requirement of the PLR in a router is related to the number of interfaces n, and the complexity is $O(n \times len_Q)$.

D. Proactive Loss Sensing Module

PLSM is the entrance of a PLR instance and is designed as a packet scheduler. When a packet arrives from the forwarder, PLSM first acquires the queue length of the egress through NMM and then determines if a packet is lost based on whether the queue is full. PLSM dispatches the lost packets to the pipeline defined by PLR instead of the regular one that directly sends them. In PLR’s pipeline, the packet will be processed differently depending on the packet type:

- **Data:** PLSM stores the packet in RS and constructs an ELN to notify the loss;
- **Interest:** PLSM only stores the packet in RS without triggering any ELN feedback;
- **ELN:** PLSM just discards the packet.

This differentiated treatment is based on the overhead and underlying traffic patterns. The ELN should return to the consumer who sent or requested the lost packet. If PLSM detects a lost Data packet on certain egress, the constructed ELN could be directly inserted into the queue of the same egress because they have the same destination. However, if PLSM wants to send an ELN to notify a lost Interest packet, it needs to search PIT for the incoming interface of the Interest packet, thus making the feedback costly. And due to the huge difference in size between Interest and Data packets, the paths that experience congestion are mainly downlink, with traffic consisting mainly of Data packets, while the upstream paths dominated by Interest packets are less likely to be congested. Considering the tradeoff between solution goals and overhead, we only choose to perform the ELN feedback on Data packets. Moreover, it should be noted that the ELN is constructed after the packet is successfully stored, which limits the number of ELN packets not to exceed the maximum size of RS determined by Eq. (1) when congestion occurs to prevent preempting too many forwarding resources.

E. Loss Recovery Module

We enable PLR to proactively send packets to retransmit lost packets stored in the RS, injecting additional traffic into the network. Considering the performance of PLR and the overall network, the additional traffic should be large enough to quickly recover the loss before expiration while not affecting the regular transmission traffic. Therefore, PLR triggers the recovery and adaptively controls the retransmission rate according to the congestion status.

Inspired by some in-network congestion detection method [11], [15], we use the local queue length as the congestion status. The basic idea is that the difference between the enqueue rate ($rate_{enq}$) and dequeue rate ($rate_{deq}$) can be revealed by the change of queue length ($qlen$) as follows:

$$\frac{\mathrm{d}qlen(t)}{\mathrm{d}t} = rate_{enq}(t) - rate_{deq}(t), \text{ if } \forall t, qlen(t) > 0,$$

where the variables are functions of t. If $\frac{\mathrm{d}qlen(t)}{\mathrm{d}t} < 0$, it means that any additional rate lower than $-\frac{\mathrm{d}qlen(t)}{\mathrm{d}t}$ will not change the trend of decreasing. The theoretical formula can be rewritten to the practical form by introducing the time interval $T_{interval}$ of counting and network parameters as follows:

$$\frac{qlen_i - qlen_{i-1}}{T_{interval}} = \frac{sendN_i - sendN_{i-1}}{T_{interval}} - \frac{BW}{MSS}, \quad \text{if } qlen_i \neq 0 \text{ or } qlen_{i-1} \neq 0, \tag{2}$$

where BW is the interface's bandwidth, and $sendN_i$ and $qlen_i$ are the total number of sent packets and the number of packets in queue counted at the end of round i, respectively. MSS is the max size of a packet.

When the condition of Eq. (2) is satisfied, we can calculate the change of $qlen$ in a round. If the $qlen$ decreases, it means that we can send additional $qlen_{i-1} - qlen_i$ packets without exceeding the bandwidth, which is the maximum available retransmission rate. And if $qlen$ increases, the current rate has exceeded the bandwidth, and no additional traffic should be injected into the network. A cumulative value R_i is introduced to represent the number of packets that can be retransmitted in round i, and it can be calculated as follows:

$$R_i = \max\left\{0, R_{i-1} + \frac{qlen_{i-1} - qlen_i}{2}\right\}. \tag{3}$$

R will increase at half the average rate at which $qlen$ decreases, and once it reaches 1, the LRM will retransmit a packet. When $qlen$ keeps increasing or jitters within a small range around a certain value, the value of R will not be greater than 1, thereby suppressing retransmission. At the same time, define N_{bw} as the maximum number of packets that the interface can send in a round, and we can estimate N_{bw} according to Eq. (2):

$$N_{bw} = sendN_i - sendN_{i-1} + qlen_{i-1} - qlen_i. \tag{4}$$

If the condition is not satisfied, we cannot use Eq. (3) to calculate the value R because $qlen$ is always zero. But there is still bandwidth available to retransmit packets in this case because the enqueue rate is definitely less than the dequeue rate. In this situation, the number of additional packets that we can send in round j is as follows:

$$R_j = R_{j-1} + \frac{N_{bw} - sendN_j + sendN_{j-1}}{2}. \tag{5}$$

The rationale of the equation is quite straightforward. N_{bw} and $sendN_j - sendN_{j-1}$ are the bandwidth we estimated and the send rate in round j, which represent the enqueue rate and the dequeue rate, respectively.

Algorithm 1: Adaptive loss recovery

```
Data: queue length $qlen_{last}$, the total number of sent
      packets $sendN_{last}$, estimated bandwidth $N_{bw}$,
      cumulative retransmission rate $R$;
1  initialization;
2  for every $T_{interval}$ do
3      $qlen_{cur} \leftarrow$ GetQlenByNMM();
4      $sendN_{cur} \leftarrow$ GetTotalSendNum();
5      if $qlen_{cur} < \tau \cdot len_Q$ then
6          $sendN_{\Delta} = sendN_{cur} - sendN_{last}$;
7          if $qlen_{last} > 0$ or $qlen_{cur} > 0$ then
8              $R \leftarrow \max\left\{0, R + \frac{qlen_{last} - qlen_{cur}}{2}\right\}$;
9              $N_{bw} \leftarrow sendN_{\Delta} + qlen_{last} - qlen_{cur}$;
10         else
11             $R \leftarrow \max\left\{0, R + \frac{N_{bw} - sendN_{\Delta}}{2}\right\}$;
12         end
13         if RS is not empty and $R > 1$ then
14             Move $\lfloor R \rfloor$ packets to the queue;
15             $qlen_{cur} \leftarrow qlen_{cur} + \lfloor R \rfloor$;
16             $R \leftarrow R - \lfloor R \rfloor$;
17         end
18     end
19     $qlen_{last} \leftarrow qlen_{cur}, sendN_{last} \leftarrow sendN_{cur}$;
20 end
```

The complete adaptive loss recovery is presented in Algorithm 1, which runs every $T_{interval}$ to implement the principles shown in Eq. (3)-(5). The queue length and the total number of sent packets are obtained through the NMM and the statistics that any modern forwarder [22], [23] will provide. We introduce a threshold $\tau \cdot len_Q$ to suppress the data update and retransmission for the reason that the calculation of $qlen_{last} - qlen_{cur}$ and N_{bw} may be inaccurate when the queue length is large. The inaccurate value may lead to an excessively large retransmission rate, which will seriously affect regular transmission, especially in this situation. The retransmission is triggered on-demand when the RS is not empty, and LRM retransmits stored packets by directly moving them to the queue. Therefore, the counted queue length should be updated to reflect the actual length, and the cumulative retransmission rate also decreases to reflect the use of retransmission resources.

The complexity of each round of this algorithm is $O(1)$, so the overall complexity is determined by $T_{interval}$. When there is no need to trigger retransmissions, the algorithm only performs basic data updates with low overhead.

V. Discussion on PLR

In this section, we analyze the behaviors of PLR and conventional loss recovery scheme about detection and recovery time to demonstrate the effectiveness and advantages of PLR.

Incomplete data should be buffered by a consumer waiting for the remaining data to construct complete content. So before the lost packet is recovered, the buffered data size will continuously grow. The buffer data size of a consumer when packet j is lost can be calculated as Eq. (6):

$$\begin{aligned} bufdata &= thruput \times AccumTime_j, \\ AccumTime_j &= t^j_{recv} - t^{j+1}_{recv}, \end{aligned} \tag{6}$$

where $AccumTime_j$ is the buffer accumulation time due to lost packet j and $thruput$ is the average throughput during this period. t^j_{recv} and t^{j+1}_{recv} are the arrival time of the lost packet j and the next packet $j+1$, respectively. In this case, if the consumer has a buffer with limited size ($bufsize$), the transmission has to be blocked when incomplete data exhausts the buffer. The blocked time $BlockTime_j$ is easy to calculate by deforming Eq. (6) as follows:

$$BlockTime_j = \max\left\{0, AccumTime_j - \frac{bufsize}{thruput}\right\}.$$

Therefore, $AccumTime_j$ plays an important role in transmission performance, and we will analyze the $AccumTime_j$ of the conventional scheme and PLR below.

In Fig. 5, we illustrate both the transmission timing diagrams of the conventional scheme and PLR. According to the analysis and the information in the figure, we can rewrite $AccumTime_j$ in Eq. (6) as follows:

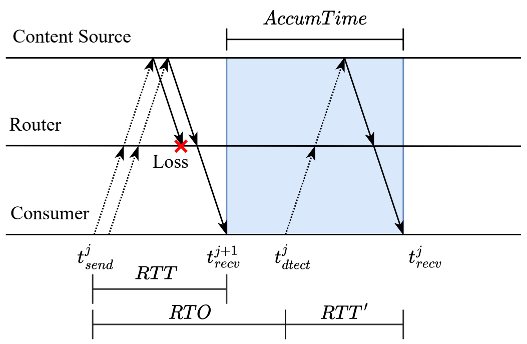
$$\begin{aligned} &AccumTime_j = t^j_{recv} - t^{j+1}_{recv} = \\ &\left(t^j_{dtect} - t^j_{send}\right) + \left(t^j_{recv} - t^j_{dtect}\right) - \left(t^{j+1}_{recv} - t^j_{send}\right), \end{aligned} \tag{7}$$

where t^j_{dtect} and t^j_{send} is the time when the consumer detects the loss of packet j and when the consumer first sends the Interest of packet j, respectively. In this form, $AccumTime_j$ is determined by three parts: detection delay and recovery delay of lost packet j, and the retrieval delay of next packet $j+1$. The above parameters of the two schemes can be easily calculated according to their principle of detecting and recovering the loss.

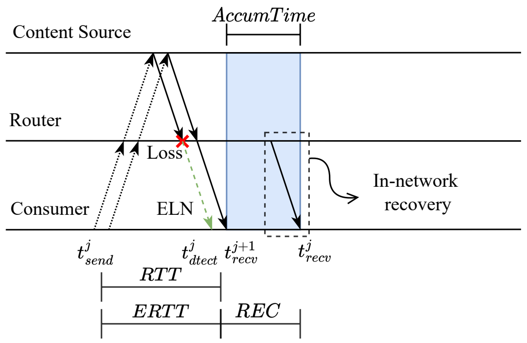
1) Conventional scheme: It uses a RTO timer to detect the loss and recover it by retransmitting the Interest. Therefore, the $AccumTime_j$ of conventional scheme can be determined as follows:

$$AccumTime_j = RTO_j + RTT'_j - RTT_{j+1},$$

where RTO_j is retransmission timeout of packet j, and RTT'_j and RTT_{j+1} are round-trip time of recovered packet j and packet $j+1$, respectively.



(a) Conventional scheme.



(b) Proposed PLR.

Fig. 5: Transmission timing diagrams of the conventional scheme and proposed PLR.

2) PLR: By design, PLR detects loss in-network and notifies consumers by ELN, whose behavior is similar to Data packets. It also recovers loss in-network when congestion mitigates. Thus, we can calculate $AccumTime_j$ of proposed PLR as follows:

$$AccumTime_j = ERTT_j + REC_j - RTT_{j+1} \approx REC_j,$$

where $ERTT_j$ is the round-trip time of packet j calculated using the arrival time of ELN and REC_j is the recovery time of the lost packet. Because of the ELN's similar behaviors to the Data packet, $ERTT_j$ is almost the same as ordinary RTT_j. So $AccumTime_j$ of PLR is determined by REC_j, which in turn depends on how fast the congestion mitigates.

As we analyzed in Section II-B, the loss detection time of conventional scheme RTO_j is usually larger than that of PLR $ERTT_j$, which results in a slower downstream response to the congestion signal. Meanwhile, RTT'_j is usually different from RTT_{j+1}, which leads to large and jittery $AccumTime_j$ of the conventional scheme. Therefore, it is easy for incomplete data to exhaust the buffer and block transmission when using the conventional scheme. On the contrary, REC_j is directly related to network conditions, but in most cases, it is much less than RTO, which has a large minimum value in practice [18]. Therefore, PLR provides faster loss detection and recovery and has a lower probability of suffering transmission blocking, thus making a higher transmission performance.

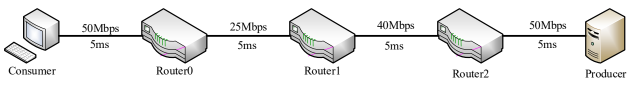
VI. Performance Evaluation

A. Environment Setup

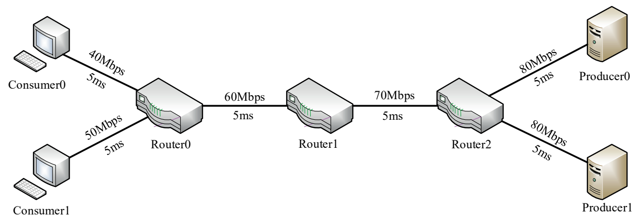
We implement our scheme PLR in the NFD and evaluate it through ndnSIM [24], an ns-3 based NDN simulator running the code of NFD directly. We compare the performance of existing congestion control algorithms with and without PLR. We choose four congestion control algorithms for testing, which are ICP [9], BIC [25], CUBIC [26] and PCON [10]. ICP represents the algorithm that uses the loss detected by RTO as a congestion signal. We also choose a variant of ICP that uses the aggressive BIC algorithm instead of AIMD as the window adjustment strategy to test the PLR under severe congestion conditions. PCON represents another type of congestion control algorithms that do not use the loss as the congestion signal. Considering that PCON is based on CUBIC, we also choose CUBIC for comparison to exclude its effect. All algorithms are already implemented in ndnSIM, and we will use their default settings.

We measure the effectiveness of PLR by three metrics: (i) completion time, which reflects the throughput benefiting from fast loss detection and notification; (ii) the number of retransmissions on the consumer, which shows the number of packets recovered before RTO timeouts, reflecting the PLR's fast in-network recovery capability; (iii) the full delay, which is the time between when an Interest packet is first sent and when the Data packet comes back.

We consider two typical network scenarios whose topologies are shown in Fig. 6. The first is the single-path transport scenario in the linear topology shown in Fig. 6(a). This simple scenario can show how a congestion control algorithm runs, which is enough to evaluate the primary effectiveness of PLR. In the dumbbell topology in Fig. 6(b), we consider a scenario that a short burst transport competes with a long regular transport. In this scenario, a consumer requests content under the control of congestion control algorithms, and the other requests content in bursts from time to time.



(a) The linear topology.

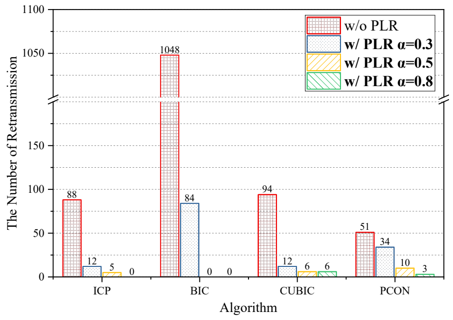


(b) The dumbbell topology.

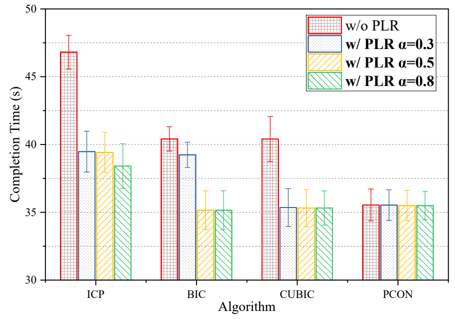
Fig. 6: The topologies for testing.

B. The Single-path Scenario

In this scenario, the consumer will acquire 100,000 Data packets with a size of 1040 bytes each under the control



(a) The number of retransmissions.



(b) Completion time.

Fig. 7: Transmission performance of different algorithms when $len_Q = 32$ and α varies in single-path scenario.

of different congestion control algorithms. The parameter $T_{interval}$ is set to 1 ms, and τ is set to 0.75, which will be unchanged in the simulation. And we choose a relatively small value 32 for the parameter len_Q to test PLR under frequent congestion conditions and evaluate the effect of PLR when parameter α changes.

The testing result is shown in Fig. 7, where "w/o PLR" represents an algorithm without PLR and "w/ PLR" represents that with PLR. In Fig. 7(a), we show the number of retransmissions of different algorithms and the trend of it when the parameter α is increasing. Without the deployment of PLR, all algorithms have many retransmissions, which represents packet loss in this situation. BIC has the largest number of retransmissions because of its overly aggressive window adjustment strategy using exponentially increasing. We can find that the number of retransmissions reduces significantly when these algorithms work with PLR, and as the parameter α increases, it reduces more until almost zero. In the best case, the number of retransmissions of all algorithms decreases by at least 90%. As a result of the retransmissions reduction, as shown in Fig. 7(b), the completion time of ICP, BIC, and CUBIC decreases by at most 21.8%, 14.9%, and 14.4%, respectively. Considering the full delay, the average of ICP, BIC, CUBIC, and PCON are 44.05 ms, 50.79 ms, 42.61 ms, and 45.08 ms, respectively, which are almost the same when they work with PLR.

Combining the above results, PLR can reduce the number of Interest retransmissions and the completion time and has little impact on delay. Now we can take a deeper discussion on the critical parameter α. If we observe the trend of these results, we can see that the effect of PLR improves with increasing α. However, the improvement achieves at least 80% at the point of 0.5, and it is relatively less when α is greater. Moreover, a larger α leads to more space requirement of RS, more ELNs, and more retransmission attempts that make PLR costly. To ensure the effectiveness of PLR while reducing the overhead as much as possible, we will set parameter α to 0.5 in default in the following evaluation.

C. Comparison Among Different Settings

In the previous section, we show the effectiveness of PLR under the condition of short queue length. To prove further that PLR is a valuable scheme, we compare the performance of all algorithms with and without PLR when len_Q increases.

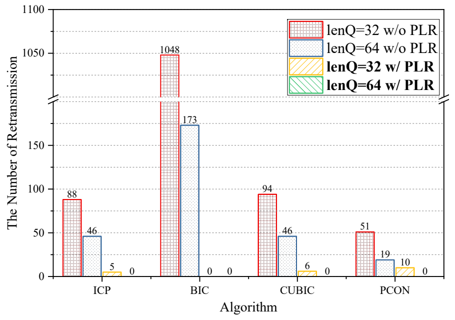
The performance comparison shows in Fig. 8(a) and Fig. 8(b). We can see that the number of retransmissions and completion time decrease as the len_Q grows from 32 to 64. Furthermore, with the use of PLR, the performance boosts much more. Comparing the result between the baseline ($len_Q = 32$ w/o PLR) and three cases with different evaluation settings, we can observe that the average reduction in the number of retransmissions in case 1 ($len_Q = 64$ w/o PLR) is only 60.5% and that in case 2 ($len_Q = 32$ w/ PLR) reaches up to 91.75%, and it is close to the best 100% in case 3 ($len_Q = 64$ w/ PLR). As for the completion time shown in Fig. 8(b), the average reduction in three cases compared with baseline is 5.38%, 10.36%, and 13.79%, respectively.

Based on the results, we can conclude that PLR is very effective in fast loss recovery for the less number of retransmissions and improving transmission performance in the single-path scenario. Moreover, we can get more performance boosts if deploying PLR instead of just extending the queue. However, although the packet loss is recovered quickly, the completion time of PCON remains hardly unchanged whether we deploy PLR or not. It is because PCON uses early feedback marking as a congestion signal, which is nearly optimal in this scenario. In the next section, we will show that the PCON is subject to the burst, and PLR also takes effect.

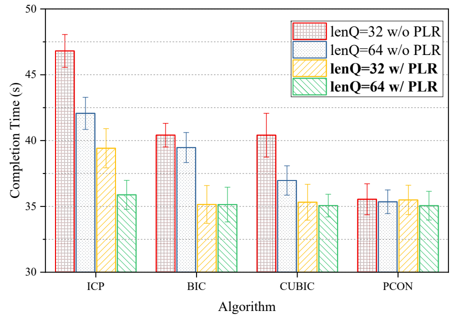
D. The Burst Scenarios

In this scenario, we consider a network with burst transmissions shown in Fig. 6(b). Consumer0 requests 50 MB of content from Producer0 under the control of different algorithms, and Consumer1 requests the content from Producer1 at a fixed rate of 35 Mbps for 0.5 s without any congestion control at time points 2.5 s, 5.0 s, and 10.0 s, respectively. In this situation, the congestion loss can occur in Router1 and Router2 because the possible highest rate is up to 75 Mbps. According to the discussion above, we use $len_Q = 64$ to get better network performance, and we set α to 0.5 to balance the effectiveness and overhead of PLR.

The result in Fig. 9(a) shows that both Consumer0 and Consumer1 have retransmissions no matter which algorithm we use. In this case, PLR has a larger number of retransmissions than BIC, which is because PCON ignores all subsequent congestion signals for a short time once reducing the window, maintaining a high throughput. PLR can also reduce the Interest retransmissions for two consumers significantly. Besides, it decreases the completion time of Consumer0 by 4.1%, 19.7%, 8.5%, and 8.6% for ICP, BIC, CUBIC, and PCON, respectively, as shown in Fig. 9(b). Burst transmissions occur in a short time and are generally delay-sensitive, so we are concerned with their delay rather than completion time. As we can see in Fig. 9(c), the average full delay of Consumer0 decreases using all four algorithms, and that of Consumer1 has a bit of difference. When Consumer0 uses BIC and CUBIC with PLR, the delay of Consumer1 increases because these two algorithms are aggressive in terms of the rate increase, and PLR slows down the reduction in send queue length during retransmission. However, the increase in delay is no more than 3 ms, and it is worthwhile if we consider the performance of the entire network in this scenario.



(a) The number of retransmissions.

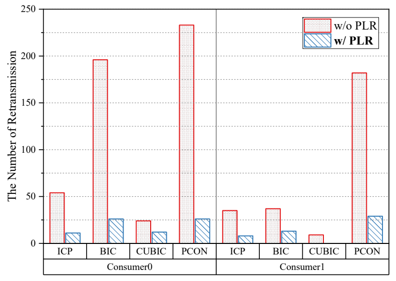


(b) Completion time.

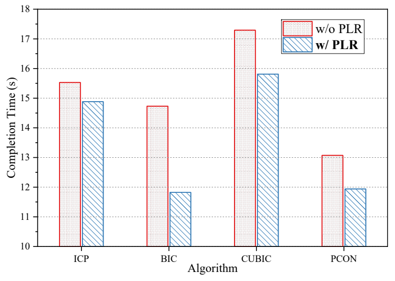
Fig. 8: Transmission performance of different algorithms when len_Q varies and $\alpha = 0.5$ in single-path scenario.

VII. Conclusion

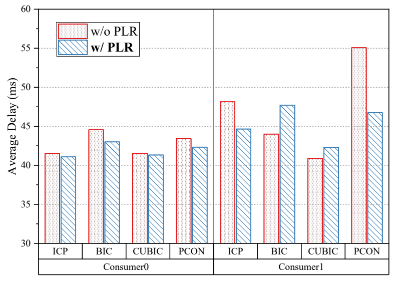
In this paper, we proposed an efficient, reliable transmission scheme called PLR, customized for NDN. Deployed on each router, PLR can timely detect and recover the congestion-induced loss. Specifically, it detects loss by monitoring the queue status of each interface and uses high-priority explicit notifications to notify consumers of the loss events. PLR stores lost packets in the router's cache and then retransmits them at an adaptive rate according to the detected remaining bandwidth. We implemented PLR in ndnSIM and verified it through simulation. The results show that PLR can significantly reduce



(a) The number of retransmission.



(b) Completion time of Consumer0.



(c) Average delay.

Fig. 9: Transmission performance of different algorithms when $len_Q = 64$ and $\alpha = 0.5$ in burst scenario.

the number of retransmissions on consumers and decrease the content completion time by up to 21.8% compared with the baseline.

Acknowledgment

The work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61972371, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] "Ericsson mobility report," accessed: Feb., 2023. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[4] E. Blanton, V. Paxson, and M. Allman, "TCP congestion control," 2009, RFC 5681, accessed: Feb., 2023. [Online]. Available: https://www.ietf.org/rfc/rfc5681.txt

[5] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking–A survey," *Computer Communications*, vol. 86, pp. 1–11, 2016.

[6] S. Vusirikala, S. Mastorakis, A. Afanasyev, and L. Zhang, "Hop-by-hop best effort link layer reliability in named data networking," University of California at Los Angeles, Tech. Rep. NDN-0041, 2016, accessed: Feb., 2023. [Online]. Available: https://named-data.net/wp-content/uploads/2016/08/ndn-0041-1-hop-by-hop-link-reliability.pdf

[7] J. Qin, K. Xue, Y. Chen, W. Wei, J. Liu, and H. Yue, "STNDN: link aware segmented transmission for named data networking," in *Proceedings of the 2019 International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, 2019, pp. 50–55.

[8] K. Matsuzono, H. Asaeda, and T. Turletti, "Low latency low loss streaming using in-network coding and caching," in *Proceedings of the 2017 Conference on Computer Communications (INFOCOM)*. IEEE, 2017, pp. 1–9.

[9] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *Proceedings of the 2012 Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2012, pp. 304–309.

[10] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for named data networking," in *Proceedings of the 2016 Conference on Information-Centric Networking (ACM-ICN)*. ACM, 2016, pp. 21–30.

[11] J. Yang, Y. Chen, K. Xue, J. Han, J. Li, D. S. Wei, Q. Sun, and J. Lu, "IEACC: an intelligent edge-aided congestion control scheme for named data networking with deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4932–4947, 2022.

[12] Y. Hu, C. Serban, L. Wang, A. Afanasyev, and L. Zhang, "BBR-inspired congestion control for data fetching over NDN," in *Proceedings of the 2021 Military Communications Conference (MILCOM)*. IEEE, 2021, pp. 426–431.

[13] S. N. S. Hashemi and A. Bohlooli, "3CP: coordinated congestion control protocol for named-data networking," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3918–3932, 2021.

[14] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, "An empirical study of receiver-based aimd flow-control strategies for CCN," in *Proceedings of the 2013 International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2013, pp. 1–8.

[15] S. Song and L. Zhang, "Effective ndn congestion control based on queue size feedback," in *Proceedings of the 2022 Conference on Information-Centric Networking (ACM-ICN)*. ACM, 2022, pp. 11–21.

[16] F. Abdi, M. Ahmadi, and M. Ghanem, "LA-MDPF: A forwarding strategy based on learning automata and markov decision process in named data networking," *Future Generation Computer Systems*, vol. 134, pp. 22–39, 2022.

[17] B. Ahlgren and K.-J. Grinnemo, "ZQTRTT: a multipath scheduler for heterogeneous traffic in icns based on zero queueing time ratio," in *Proceedings of the 2022 Conference on Information-Centric Networking (ACM-ICN)*. ACM, 2022, pp. 1–10.

[18] M. Sargent, J. Chu, D. V. Paxson, and M. Allman, "Computing TCP's Retransmission Timer," 2011, RFC 6298, accessed: Feb., 2023. [Online]. Available: https://www.ietf.org/rfc/rfc6298.txt

[19] G. Carofiglio, L. Muscariello, M. Papalini, N. Rozhnova, and X. Zeng, "Leveraging icn in-network control for loss detection and recovery in wireless mobile networks," in *Proceedings of the 2016 Conference on Information-Centric Networking (ACM-ICN)*. ACM, 2016, pp. 50–59.

[20] K. Ueda, T. Kato, C. Sasaki, and A. Tagami, "Revisiting loss detection in ndn: Detecting spurious timeout using probe interest," in *Proceedings of the 2022 Global Communications Conference (GLOBECOM)*. IEEE, 2022, pp. 4455–4460.

[21] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *Proceedings of the 2013 Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2013, pp. 363–368.

[22] J. Shi, D. Pesavento, and L. Benmohamed, "NDN-DPDK: Ndn forwarding at 100 gbps on commodity hardware," in *Proceedings of the 2020 Conference on Information-Centric Networking (ACM-ICN)*. ACM, 2020, pp. 30–40.

[23] J. Takemasa, Y. Koizumi, and T. Hasegawa, "Vision: toward 10 tbps ndn forwarding with billion prefixes by programmable switches," in *Proceedings of the 2021 Conference on Information-Centric Networking (ACM-ICN)*. ACM, 2021, pp. 13–19.

[24] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnSIM: an open-source simulator for NDN experimentation," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.

[25] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proceedings of the 2004 Conference on Computer Communications (INFOCOM)*, vol. 4. IEEE, 2004, pp. 2514–2524.

[26] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.