FFRD: Fragment Forwarding and Reassembly Decoupling based Chunk Transmission in NDN

Chengbao Cao[1], Kaiping Xue[1*], Hao Yue[2], Junjie Xu[1]
[1] Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
[2] Department of Computer Science, San Francisco State University, San Francisco, CA 94132, USA
*kpxue@ustc.edu.cn

***Abstract*—In-network caching is an inherent feature of Named Data Networking (NDN), and the basic data unit of naming and caching in NDN is called "chunk". However, a chunk needs to be further fragmented into fragments when its size is larger than the link layer's Maximum Transmission Unit (MTU). Furthermore, fragments also need to be reassembled into the original chunk at intermediate routers so that subsequent requests can be satisfied by the cached copy. The current NDN design adopts a coupled hop-by-hop reassembly mechanism where fragments can be forwarded to the next hop only if the chunk has been fully reassembled, which leads to a significant end-to-end delay when large chunks are transmitted due to the processing delay at intermediate routers. In this paper, we propose a reliable and fast chunk transmission protocol based on Fragment Forwarding and Reassembly Decoupling (FFRD) at intermediate routers in NDN. In FFRD, fragments are forwarded to the next hop upon being received and reassembly occurs after all fragments are received. Meanwhile, FFRD can timely detect and recover packet losses at intermediate routers to minimize the transmission delay. The simulation results show that FFRD can significantly reduce chunk retrieval delay and decrease end-to-end Interest packet retransmission times, especially over lossy networks with non-negligible packet losses.**

***Index Terms*—Named Data Networking (NDN), chunk transmission, reassembly, reliability, bitmap**

I. Introduction

Traditional TCP/IP architecture was originally designed to enable communications between end hosts. However, due to the rapid growth of mobile data traffic, TCP/IP architecture is facing serious challenges in mobility and content distribution. In this situation, Named Data Networking (NDN) [1] has been considered to be applied to wireless environment in many research works [2] [3].

In the NDN model, the content objects are split into small chunks and each chunk is uniquely identified by a name (*e.g., /provider/video-name/chunk-id*). Consumers generate a request by sending out Interest packets, which contain the names of the desired chunks. Each intermediate router in the path forwards the Interest packet by looking up the *Forwarding Information Base (FIB)* and maintains a *Pending Interest Table (PIT)* to keep track of the current unsatisfied Interest packets and the corresponding ingress faces. Matched chunks will be sent back through the reverse direction of the request path to the consumer according to the PIT entries. One of the most significant features is that as the chunk is transmitted backwards to the consumer, intermediate nodes can insert the copy of each chunk into the *Content Store (CS)*, so that routers can directly reply to the subsequent Interest packets for the same chunks.

Since chunk is the basic data unit of naming and caching in NDN, one issue that has to be handled is fragmentation and reassembly of large chunks due to the limitation of the link layer's Maximum Transfer Unit (MTU) size. NDNx prototype [4] recommends the chunk size to be smaller than 8800 bytes, and many research works [5] [6] use a chunk size smaller than the MTU (*e.g., 1024bytes*) simply to avoid fragmentation. However, the small chunk size is inefficient for high-quality delay-sensitive video streaming or large file delivery, since data consumers have to send plenty of Interest packets at each time point to achieve real-time playing. For example, a data consumer of VoCCN (Voice Over CCN) has to send more than 50 Interest packets per second to guarantee user experience [7] [8]. Such high-rate Interest packets may cause network congestion and result in Interest packet and Data packet loss, especially in the wireless network. To address these issues, we need to increase the chunk size for large files or real-time high-quality video delivery.

Till now, only a few research works have investigated on how to fragment a chunk into fragments and reassemble them at intermediate routers when chunk size is larger than the MTU size. NDN Link Protocol (NDNLP) [9] is an optional mechanism currently running on NDNx prototype, which fragments each chunk into multiple MTU-compliant link layer packets after reassembled at each router hop-by-hop. In other words, each router forwards a chunk to the next hop unless all fragments of the chunk are received. In [10], Satyanarayana et al. present a Best Effort hop-by-hop Link layer Reliability Protocol (BELRP), which can detect and recover packet losses between routers. However, both methods in [9] and [10] rely on hop-by-hop reassembly, where fragments cannot be forwarded individually until they are reassembled into a complete chunk. Hence, additional time consumption for processing chunk reassembly at intermediate routers is unavoidable. Besides, Mosko et al. [11] introduced an end-to-end fragmentation protocol for CCNx [12] to avoid the transmission delay stated above. However, in [11], intermediate routers cache not chunks themselves but their fragments.

In this paper, we propose to fragment each large chunk into multiple data packets, where each data packet contains a sequence number and a complete chunk name. Intermediate routers will forward every fragment to the next hop immediate-

978-1-5386-2062-5/17/$31.00 ©2017 IEEE

ly upon receiving it. Meanwhile, a bitmap field is introduced to record the state of the chunk. When timeout occurs and the chunk is incomplete, the intermediate router will send a NACK packet with an additional bitmap field for selective retransmission between routers. Each router inserts the chunk into the CS after it is successfully reassembled. The main contributions of this paper are summarized as follows:

- We propose a chunk transmission mechanism for Named Data Networking which decouples fragments forwarding and reassembly (FFRD). In FFRD, reassembly happens at both intermediate routers and end hosts but have no effects on the end-to-end delay compared with no intermediate router's reassembly. Hence, FFRD is suitable for large chunk transmission or time-sensitive high-quality video streaming.
- A reliable transmission mechanism to reduce end-to-end Interest packet retransmission when packet loss happens is also proposed. In FFRD, packet loss detection and recovery are implemented at both intermediate routers and end hosts, where a bitmap field is introduced to assist chunk reassembly.
- We implement the FFRD in ndnSIM and conduct experiments to evaluate performance with the default NDNLP. The simulation results show that the proposed protocol can achieve its design goals.

The remainder of this paper is organized as follows. Section II presents the background and motivations of this paper. Then we illustrate the design of FFRD in Section III. Performance evaluation results are shown in Section IV. Finally, section V concludes this paper.

II. Background

Packet fragmentation is necessary for NDN architecture due to the link layer's limitation on MTU. In traditional TCP/IP architecture, TCP will segment a byte stream into IP packets and IP needs to further fragment a packet into multiple smaller packets if it is larger than the link layer MTU limitation. More specifically: 1) IPv4 [13] chooses hop-by-hop fragmentation with end-host reassembly mechanism. Each hop fragments the packet to fit MTU size if needed, but fragments are not immediately reassembled by the routers, only the final destination reassemble the packet. 2) IPv6 [14] chooses end-to-end fragmentation and reassembly mechanism. The packets are fragmented according to the path MTU size at the source but not supposed to be further re-fragmented at the intermediate routers and will only be reassembled at the destination.

However, both these schemes in IPv4/IPv6 cannot be directly applied to NDN architecture because there are some differences between NDN architecture and the traditional TCP/IP architecture:

- *Caching at routers:* Only fully reassembled chunk will be cached for the subsequent request. If fragments are not reassembled at intermediate routers, requesting or naming must be applied to the fragments, which is unrealistic.
- *Symmetric routing:* As shown in Fig. 1, fragments of the same IP packet might be transmitted along different paths

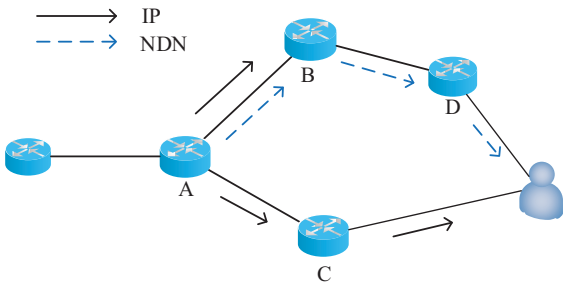

Fig. 1: IP fragments might be transmitted along different path(e.g., A → B or A → C), router D can not receive all fragments. While in NDN, all fragments are transmitted along the same path (e.g., A → B → D) according to the PIT entry

int the traditional TCP/IP architecture. However, in NDN architecture, fragments of the same NDN chunk will follow the same sequence of NDN routers, through retracing PIT state set up by a preceding interest. Therefore, each chunk can be successfully reassembled even though hop-by-hop reassembly is not applied.

Due to the above differences, NDNLP [9] is proposed, which adopts an hop-by-hop fragmentation and reassembly mechanism. In [9], routers have to receive all data fragments before transmitting the chunk to the next hop. Moreover, due to the lack of reliability assurance, one chunk's fragment loss will cause Interest retransmission by the application and hence increase time delay. It is obvious that when fragmentation is enabled, the delay caused by hop-by-hop reassembly is intolerable and a detailed delay evaluation will be shown in section IV.

Thus, in order to provide a better performance for time-sensitive applications, it is critical to propose a feasible chunk fragmentation and reassembly protocol for NDN architecture.

III. Protocol Design

In this section, we present the design of FFRD, a protocol handling chunk fragmentation and reassembly with reliability assurance in NDN. We first describe our design goals and then elaborate on the procedure of the FFRD. In this paper, we use "chunk" to refers to the basic data unit that one Interest packet brings back to the consumer, and "data packets" to refer to the fragments when a chunk is fragmented into multiple MTU-compliant packets. Data packets with the same chunk name but different sequence numbers can be forwarded individually and reassembled later. The router that receives all data packets with the same chunk name can reassemble them into a complete chunk and insert it into the content store.

A. Design Goals

Due to the NDNLP's drawbacks, the main goals of FFRD are described as follows:

Reliability: NDN applications that require reliable chunk transmission should deploy an end-to-end reliability mechanism and retransmit an Interest packet for the chunk if necessary. We aim to minimize the application's Interest packet retransmission times.

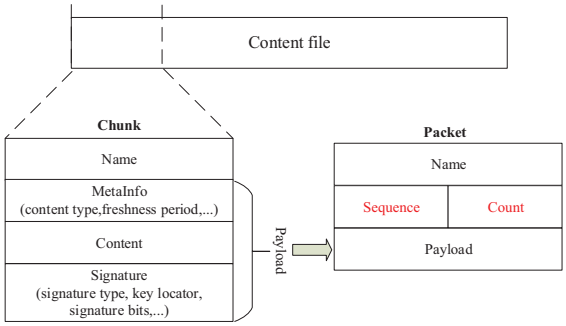

Fig. 2: Chunk and Data packet

Delay: Current hop-by-hop reassembly is a time-consuming task for the real-time applications or delay-sensitive applications. We aim to minimize the data retrieval time.

Chunk size: Small chunk size is not suitable for high-quality delay sensitive video delivery. We propose to increase the chunk size to reduce Interest packet send rate.

B. Protocol Overview

A content file requested by the consumer will be split into a number of chunks. Each chunk is composed of the payload and a header containing a unique chunk name and some utility information, such as the signature. As shown in Fig. 2, we fragment the original chunk into multiple data packets with the sequence number, total number of the chunk and chunk name in its header due to the MTU constraint. Sequence number denotes the relative position in the chunk and is also used for selective retransmission. To discover the minimum MTU size in the path (the *path MTU*), we use the method similar to [12] in CCNx 1.0 that each interest packet records the minimum MTU of links on which it has been forwarded in its header.

In order to record the receipt status of the chunk, we extend the original Pending Interest Table (PIT) entry with a bitmap field, where a bit 0 in position k means that packet k is missing, and the bit in the bitmap are initialized to zero. PIT entry will be deleted when all bits are set to 1. All data packets with same chunk name are forwarded according to the ingress face recorded in the PIT. Besides, in order to avoid redundant data packet delivery if a chunk is fragmented into multiple data packets. We propose to create a new type of NACK packet which contains the bitmap field used for selective retransmission only between two routers.

C. Protocol Detail

The detailed protocol operations when a router receives a data packet from its upstream node and a NACK packet with bitmap from its downstream node are explained below and summarized in Algorithm 1 and Fig. 3.

a) After sending an interest: Consumers send out an Interest packet for the desired chunk. Once content hit happens at intermediate router or server, following the procedure at Line 1-8, the producer fragments the chunk into multiple packets according to the minimum MTU size which is recorded in the Interest packet header.

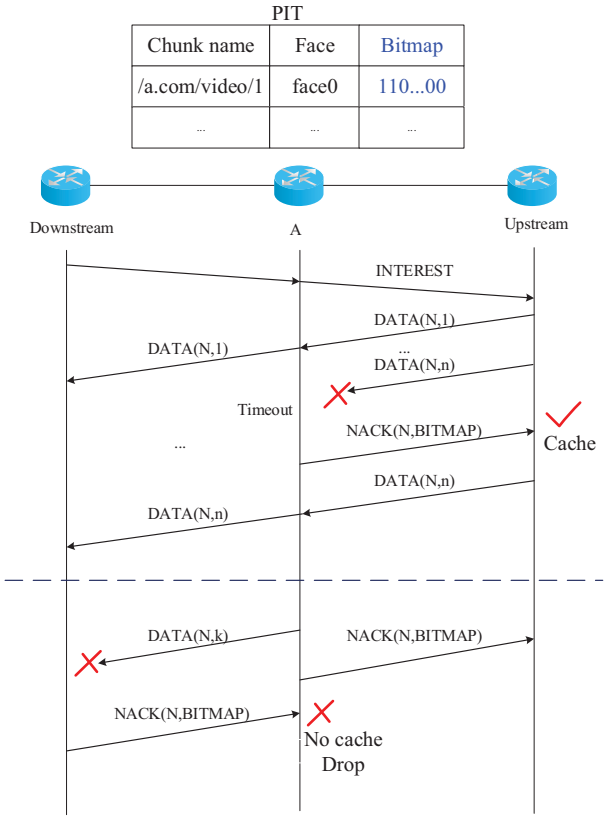

Fig. 3: Protocol Description

Meanwhile, if the related chunk is not received in a given time, the same Interest is retransmitted by the consumer. Interest packet timer is set by dynamic RTT computation. RTT is updated according to the Exponential Weighted Moving Average (EWMA) formulation when the consumer receives a complete chunk:

$$RTT_i = \beta RTT_{i-1} + (1-\beta)\overline{RTT}_i$$

where RTT_{i-1} is the average RTT estimate at the previous step, and β is a constant between 0 and 1. Like [15], $\beta = 0.85$ is proved to be good to filter out transient effects.

b) After receiving a data packet: Upon receiving a data packet (at Line 9-27), as described in Fig. 3, an NDN router A inserts the packet *D(N,1)* (*N,1* represent the chunk name and sequence number respectively) that has not been reassembled into a temporary buffer. Then it finds the matching PIT entry, sets the corresponding bit in the bitmap to 1 according to the sequence number in data packet's header and forwards the data packet to all downstream interfaces listed in PIT entry immediately. Meanwhile, the router A also checks the bitmap field and then removes the corresponding PIT entry and the temporary buffer when all bit are set to 1. Otherwise, the router updates the NACK timer and inserts the data packet into a temporary buffer space. When NACK timer expires but intermediate node A doesn't receive a complete chunk due to data packet *D(N,n)* loss, node A sends a NACK packet

D(N,BITMAP) to its upstream router for packet retransmission. Besides, in order to minimize unnecessary retransmissions due to the NACK packet loss, each NACK packet are sent twice, which increases reliability with a slight overhead increase.

A NACK timer will start when a router receives a data packet with a new chunk name, upon receiving the next packet, router calculate the ITT_{name} which represents the average inter-arrival time between two consecutive data packets. ITT_{name} is update at each data packet reception as:

$$T_{name,i} = \beta ITT_{name,i-1} + (1-\beta)\overline{ITT_{name,i}} \quad (1)$$

Then the NACK timer is set as:

$$T_i = ITT_{i-1} * N \quad (2)$$

Where N is the data packet number that the router has not been received to reassemble into a chunk. Notice that when a router receives a duplicated data packet, the NACK timer is not updated. Before the timeout, the intermediate node receives all data packets and then reassembles them into the chunk. In this case, node A cleans the corresponding PIT entry and inserts it into the content store.

c) After receiving a NACK packet: At this time, the router retransmits the lost data packets if the content store has the corresponding cache. Otherwise, the router drops the NACK packet simply since router A has already sent NACK packet and will forward all data packets to its downstream node once router A receives the missing data packet.

To avoid endless timeout due to link interrupt, the maximum attempts to recovery packet loss at intermediate routers (at Line 21) is set to be a configurable number. To fully understand the trade-off of how hard FFRD should try to restore a lost packet by the intermediate routers, we performed experiments with varying numbers of maximum retransmission times as discussed in Section IV.

D. Protocol Analysis

To support reliable chunk transmission at intermediate routers, the main cost introduced in FFRD is the timer and temporary buffer. Each sender and receiver use a temporary buffer space to store each packet which has not be reassembled. However, since each data packet can be forwarded independently, to reduce the average cost of the intermediate router in the network, it is possible to select partial routers other than all intermediate routers to perform the restore operation. More specifically, we observe that there exist many studies about caching strategy that cares about caching locations in NDN. Hence, the default caching strategy Leave Content Everywhere (LCE) considered to be replaceable. Based on this, we suggest to select the caching node that takes the responsibility for packet loss recovery. In other words, node caches content should store the data packets into a temporary buffer and maintains the corresponding timer.

Another feasible way is that the producer sends back data packets with a hop field and decreases hop-by-hop. When hop value decreases to zero, the corresponding router should store the corresponding packets in buffer space and reset hop value. The initial hop value can be determined according to hop value that interest packet carries in its head field. If the interest packet's hop value is larger than a threshold, it would be essential to set data packet's initial hop value smaller to avoid end-to-end retransmission. If intermediate routers are not selected, the router simply forwards the data packet without integrity assurance.

Without loss of generality, the node described in Fig. 3 can be treated as the node discussed above.

Algorithm 1: FFRD Protocol Detail

```
   // Interest procedure
 1 if contentHit()==true then
 2     MTU = getMTUsize(Interest)
 3     fragment(chunk,MTU)
 4     for each fragments do
 5         packets = addHeader(name,sequence,totalNum)
 6         sendToDownstream()
 7     end
 8 end
   // Data packets procedure
 9 name, sequence, totalNum = getHeader(packet)
10 sendToDownstream()
11 insertTemporaryBuffer(packet)
12 updateBitmap(sequence)
13 if all bit==1 then
14     insertCs(chunk)
15     removeFromTempBuffer(packets,name)
16 end
17 else
18     updateTimer(name)
19 end
20 if Timeout(name)==true then
       // MAXRT : maximum retransmission
          times at intermediate routes
21     if retransmission time < MAXRT then
22         sendNACK(name,bitmap)
23     end
24     else
25         removefromBuffer(packets,name)
26     end
27 end
   // NACK packet procedure
28 recvBitmap ← GetBitmap(NACK)
29 if havePacketsToSend(recvBitmap) then
30     sendPacketsToNextHop(packets)
31 end
32 drop NACK
```

IV. Performance Evaluation

In this section, to evaluate the transmission efficiency improvement including *reliability* and *end-to-end delay*, we implement our FFRD in ndnSIM [4], an NS-3 based NDN simulator. We compare the performance of the proposed

protocol with the default NDNLP [9].

Firstly, we use a linear topology with max 5 hop to examine the effect of hop length, chunk size and intermediate router's maximum retransmission times influences on reliability and end-to-end delay. The parameters used in this simulation are shown in Table I.

TABLE I: Simulation Parameters

Simulation parameters	value
Link capacity	10Mbps
Propagation	10ms
Interest number	2000
File size	4MB

To calculate the average chunk retrieval time, in this s-cenario, we use the following request model: an application sends one Interest packet and waits for the desired chunk, the application sends next Interest packet only if it has received a complete chunk.

A. Data retrieval time

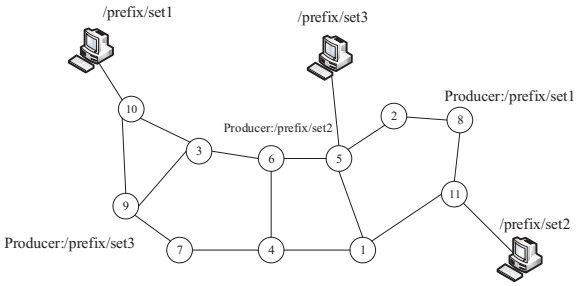

Fig. 4: Abilene topology

Firstly, we evaluate the average chunk delay when hop length is 5 and no packet loss over each link, the result is described in Fig. 5. When chunk size increase, the differ-ence of average chunk delay between NDNLP and FFRD is obvious. Considering that a large chunk has to split into multiple fragments and reassembled hop-by-hop in NDNLP, however, FFRD decoupling the fragments forwarding and reassembling at the intermediate router. Hence, the difference between NDNLP and FFRD is mainly due to the hop-by-hop reassembly. Besides, the chunk size is an important factor when calculating the data retrieval time. As show in Fig. 6, with different loss rate over each link, large chunk retrieval time is longer especially with high packet loss rate, the average chunk delay is two times when packets loss rate at 0.05 compared with no packet loss. Hence, we recommend using small chunk size especially for the real-time application which cares more about average one chunk delay but not total time downloading a file.

Furthermore, Fig. 7 describes the file download time with d-ifferent chunk size and hop length at loss rate 0.01. The results show that both NDNLP and FFRD decrease file retrieval time when chunk size increase. However, increasing the hop length, the gap between NDNLP-4KB, FFRD-4KB and NDNLP-8KB, FFRD-8KB is gradually increased. Such a behaviour is due to the fact that NDNLP is a hop-by-hop reassembly mechanism, along with hop increase, the time spends on the reassembly at intermediate routers is increasing. Secondly, large chunk size has to split into fragments, and one fragments loss will cause entire chunk drop in NDNLP. However, because of the retransmissions at intermediate routers, the time cost is increased slightly.

B. End-to-end retransmissions

End-to-end retransmissions are triggered when interest packet is lost or the corresponding chunk packet is lost. Fig. 8 shows the comparison of application retransmissions times between NDNLP and FFRD with various packet loss rate over each link. Fig. 8 shows that NDNLP is sensitive with loss rate, when loss rate increase, the end-to-end retransmissions increase rapidly. For example, If loss rate at each link is 0.03, NDNLP's retransmission times is nearly 700 and FFRD's retransmission times is nearly 200, which decrease nearly 70% retransmissions. The reason is obvious: one fragments loss will cause entire chunk drop in NDNLP and end-to-end retransmission is unavoidable. On the contrary, FFRD utilizes intermediate routers restore the lost packet, hence significantly reduce end-to-end retransmission times.

For the second scenario, we use the Abilene topology as depicted in Fig. 4 to study the impact of chunk size in the real scenario on reliability for the proposed protocol. Three consumers, at node 10, 11, 5 respectively, request for a 2MB size file(e.g.,/prefix/set1, /prefix/set2, /prefix/set3). Each link propagation delay is 5ms, and the max queue size at each node is set to 80.

C. Intermediate routers retransmissions

Below we examine the effect of intermediate routers' max retransmission times on end-to-end reliability. Given the same chunk size, with different recovery attempt times at interme-diate routers, we calculate the total time of downloading a 4M file, the results are shown in Fig. 9. It is obvious that both NDNLP and FFRD retrieval time increase, but due to the router's attempt to recover from the loss, time increased slightly in FFRD. We also observe that compared with no intermediate route's recovery, nearly 33% time decrease can be achieved when loss rate over each link is set to 0.05 and max retransmission times are set to 1. However, slightly time decrease when max retransmission times are set to 2 or 3 compared with max retransmission time set to 1. In other words, intermediate routers send back one NACK packet with bitmap can nearly restore all the lost data packets.

Meanwhile, as shown in Fig. 10, large chunks can decrease file download time to a certain degree, however, it is important to note that the chunk size is not the larger the better. In this scenario, when chunk is larger than 32KB, node 11's download time is slightly increase. The reason is: large chunk size will set longer Interest retransmission time, and if not being reassembled before the timeout, the interest needs to be retransmitted, hence increase file download time.

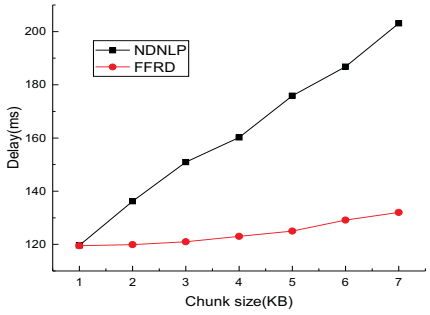

Fig. 5: Average chunk delay between NDNLP and FFRD

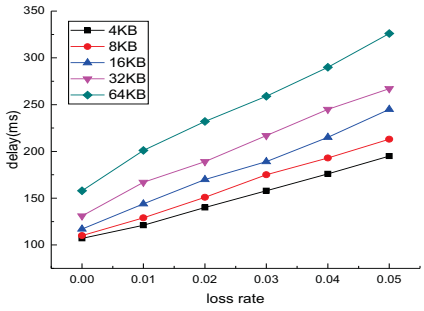

Fig. 6: Average chunk delay with different chunk size

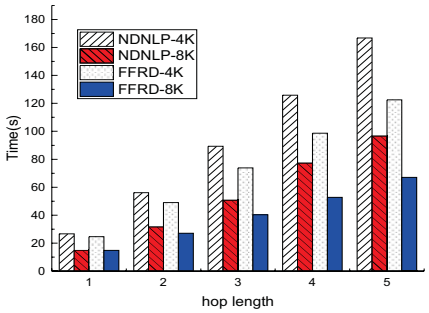

Fig. 7: File download time with variable hop length

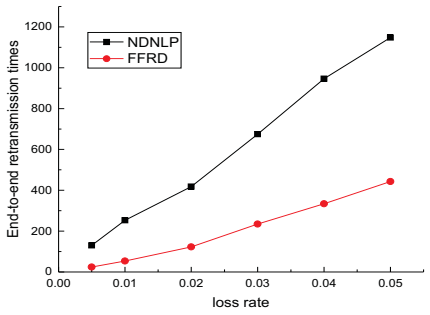

Fig. 8: End-to-end-retransmission times

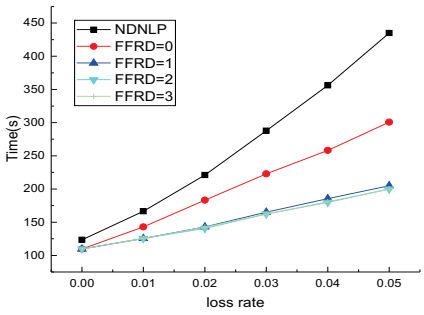

Fig. 9: Max retransmissions at intermediate routers

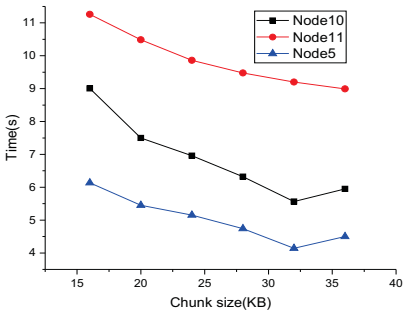

Fig. 10: File download time with different chunk size

V. Conclusion

In this paper, the issue of reliable large chunk transmit has been considered. The proposed protocol FFRD realizes larger chunk delivery compared with the default NDNLP protocol for the delay-sensitive high-quality video streaming. Because there is no necessity for routers to perform hop-by-hop reassembly, the end-to-end delay decreases significantly. Simultaneously, our simulation results demonstrate that one attempt for packets restore at intermediate routers can lead to notable performance gains for applications and decrease end-to-end interest retransmissions. Besides, with different packet loss rate, we evaluate the influence of different chunk size on end-to-end delay, the result shows that time increase faster with larger chunk size when packet loss rate increase.

Acknowledgement

This work is supported in part by the National Key R&D Program of China under Grant No. 2016YFB0800301, the National Natural Science Foundation of China under Grant No. 61379129 and No. 61671420, Youth Innovation Promotion Association CAS, and the Fundamental Research Funds for the Central Universities.

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] M. Amadeo, C. Campolo, and A. Molinaro, "Forwarding strategies in named data wireless ad hoc networks: Design and evaluation," *Journal of Network and Computer Applications*, vol. 50, pp. 148–158, 2015.

[3] B. Han, X. Wang, N. Choi, T. Kwon, and Y. Choi, "AMVS-NDN: Adaptive mobile video streaming and sharing in wireless named data networking," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 2013, pp. 375–380.

[4] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," *NDN, Technical Report NDN-0028*, 2015.

[5] J. Zhou, Q. Wu, Z. Li *et al.*, "A proactive transport mechanism with explicit congestion notification for ndn," in *Proceedings of 2015 IEEE International Conference on Communications (ICC 2015)*. IEEE, 2015, pp. 5242–5247.

[6] N. Rozhnova and S. Fdida, "An extended hop-by-hop interest shaping mechanism for content-centric networking," in *Proceedings of 2014 IEEE Global Communications Conference (GLOBECOM 2014)*. IEEE, 2014, pp. 1–7.

[7] V. Jacobson, D. K. Smetters, N. H. Briggs *et al.*, "VoCCN: voice-over content-centric networks," in *Proceedings of the 2009 Workshop on Re-architecting the Internet (ReArch '09)*. ACM, 2009, pp. 1–6.

[8] X. Jiang and J. Bi, "Interest set mechanism to improve the transport of named data networking," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 515–516.

[9] J. Shi and B. Zhang, "NNDLP: A link protocol for NDN," *The University of Arizona, Tucson, AZ, NDN Technical Report NDN-0006*, 2012.

[10] V. Satyanarayana, A. A. Mastorakis Spyridon, and L. Zhang, "Hop-by-hop best effort link layer reliability in named data networking," *University of California, Los Angeles, NDN Technical Report NDN-0041*, 2016.

[11] M. Mosko and C. A. Wood, "Secure fragmentation for content centric networking," in *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*. IEEE, 2015, pp. 506–512.

[12] M. Mosko, I. Solis, E. Uzun, and C. Wood, "CCNx 1.0 protocol architecture," *Technical Report*, 2015.

[13] J. Postel, "Internet protocol," *IETF RFC 791*, 1981.

[14] S. E. Deering, "Internet protocol, version 6 (IPv6) specification," *IETF RFC 2460*, 1998.

[15] M. Amadeo, C. Campolo, and A. Molinaro, "Design and analysis of a transport-level solution for content-centric vanets," in *Proceedings of 2013 IEEE International Conference on Communications (ICC) Workshop on Emerging Vehicular Networks: V2V/V2I and Railroad Communications*. IEEE, 2013, pp. 532–537.