NDN, NDN Memo, Technical Report NDN-0032, 2015. http://named-data.net/techreports.html
Revision 1: May 12, 2015

Packet Fragmentation in NDN: Why NDN Uses Hop-By-Hop Fragmentation

NDN Memo

Alexander Afanasyev, UCLA, afanasev@cs.ucla.edu
Junxiao Shi, University of Arizona, shijunxiao@email.arizona.edu
Lan Wang, University of Memphis, lanwang@memphis.edu
Beichuan Zhang, University of Arizona, bzhang@arizona.edu
Lixia Zhang, UCLA, lixia@cs.ucla.edu

***Abstract*—The Named Data Networking (NDN) architecture uses application-level data as units of communication, essentially giving applications much more fine grained control over network-level operations. However given the the wide spectrum of maximum transmission unit (MTU) sizes in today and future networks: which data unit size should NDN applications use? We believe that while applications should be aware of the prevalent MTU sizes in the network, application data units should not be constrained by the smallest possible MTU size among all networks. The question is then how to handle those NDN packets that exceed the link MTU size. In this paper we argue that in the context of NDN, hop-by-hop fragmentation and reassembly (HBH-FR) is the most consistent with the data-centric and session-less nature of NDN communication. We also show that, contrary to the common belief (due to misunderstanding of an earlier study), HBH-FR provides better performance tradeoffs compared to other fragmentation options.**

I. Introduction

The Internet is made of many heterogeneous networks and the level of this heterogeneity is continuously growing. Constant increase of mobile networks, continuous penetration of the Internet connectivity into homes and vehicles, connecting remote villages and even oceans (underwater networking) all result in computing devices to be connected by all kinds of means, ranging from sonar to bluetooth to fiber. One consequence of the heterogeneity in link-layer technology is the different maximum transmission unit (MTU) sizes across different links; any new Internet architecture needs to take this factor into consideration so that applications can work across networks with different links and continue to work as link layer technologies evolve.

Named Data Networking (NDN) architecture ([1], [2], [3]) makes application data the focus of communication. This switch brings multiple advantages, including data-oriented security, built-in multicast capability, ability to cache application data and retrieve cached data regardless of the application type. At the same time, heterogeneous link-layer technology brings up a question for the architecture design: whether applications need to fit their data into some minimum MTU size, or the network should implement some form of fragmentation and reassembly.

While applications should be aware of the prevalent MTU sizes in the network so that their data sizes are not too big or too small to seriously affect application performance, we believe that **application data units should not be constrained by the smallest possible MTU size among all networks** for the following reasons. First, if an application does segments its data this way, it can be very inefficient if its packets do not actually traverse links with the smallest MTU. Second, if most of the links have a higher MTU, constraining the application data unit to one or two links with a small MTU can also be quite inefficient.

If we do not require applications to segment their data in order to fit some minimum MTU size, the question is then how to handle those packets that exceed the link MTU size. There are several options for fragmenting and reassembling packets, the first two of which have been adopted in IP networks:

1) *Hop-by-Hop Fragmentation with End-Host Reassembly* or simply ***IPv4-style Fragmentation/Reassembly***: in IPv4 [4], each hop (re)fragments the packet as needed to fit its MTU size. The fragments are not immediately assembled by the next hop, but instead are assembled at the final destination;
2) *End-to-End Fragmentation and Reassembly* or simply ***IPv6-style Fragmentation/Reassembly***: in IPv6 [5], the IP protocol at the source first discovers the smallest MTU over the path to the destination (PMTU), and then fragments data from higher layers to fit the PMTU. The packets are not supposed to be further re-fragmented en-route and will be re-assembled at the destination;
3) ***Hop-by-Hop Fragmentation and Reassembly***: each hop fragments a packet as needed and the next hop assembles the packet immediately.

We believe that IPv4-style fragmentation/Assembly is not a viable option for NDN, given that: (1) interest packets cannot be partially processed, as routers need to know the whole question in order to answer it; (2) routers have to have a full data packet in order to match it against pending interests; and (3) caches should hold full data packets, as the same data can be requested by multiple clients with disparate MTU sizes. Moreover, IPv6-style fragmentation is not viable for NDN either, because the concept of Path MTU does not apply to NDN due to caching and asynchronous consumption of data and therefore producers cannot fragment or properly presize data packets.

Due to the above reasons (see Section III for more detailed explanation), NDN chooses to perform fragmentation and reassembly on a hop-by-hop basis (Option 3 above), i.e., when a packet cannot be delivered through a link due to MTU constraints, it will be fragmented and then immediately reassembled by the next hop router for further processing.

Strictly speaking, NDN's fragmentation and reassembly functionality is performed by a shim layer below the network layer of NDN when required[1], which means that the network layer is unaware of and does not deal with fragments at all.

Note that it is infeasible in general for IP to do reassembly at the immediate next hop, since IP may deliver different fragments of the same IP packet over different paths. The consequence is that if any of the fragments is lost, the other fragments will still be delivered by the network instead of being discarded at the next hop, which leads to unnecessary consumption of network resources.

In theory, IP networks could also use a shim layer below IP to perform fragmentation and reassembly as NDN does, but the concern is that the same packet may potentially suffer from repeated fragmentation and reassembly along the path. This has also been an argument against HBH fragmentation and reassembly in NDN.

We argue that HBH fragmentation and reassembly in NDN not only ensures the correctness of interest and data packet delivery, but also brings a number of advantages (Section III), including efficient handling of fragment losses and efficient utilization of large MTU sizes when only a few links in the middle have constrained MTU. In fact, "transparent" (i.e., HBH) fragmentation and reassembly was also recommended by Kent and Mogul in their seminal paper [6], often (mis)cited as a justification for E2E-based approach.

In the rest of this memo, we first overview Kent and Mogul's findings and conclusions in "Fragmentation Considered Harmful" paper [6], and then discuss in more detail the reasoning behind NDN's decision of HBH fragmentation and reassembly.

II. A BRIEF SUMMARY OF "FRAGMENTATION CONSIDERED HARMFUL"

The original IP architecture design [4] stated that IP packet fragmentation can be performed by any IP node as needed, and reassembly is done by the end receiving nodes only. The paper by Kent and Mogul "Fragmentation Considered Harmful" [6], published in SIGCOMM 1987, is the first study on the impact of fragmentation and reassembly. The presented argument is that IP packet fragmentation is "at best a necessary evil; it can lead to poor performance or complete communication failure", suggesting that one should look into a variety of ways to reduce the likelihood of fragmentation.

A. *Why Harmful?*

Kent and Mogul stated two major performance costs of IP packet fragmentation [6]. First, fragmentation leads to inefficient use of resources:

- Not only the router performing fragmentation incurs an extra computation cost, intermediate gateways must also forward more packets. Furthermore, the receiving host must reassemble the fragments.
- Additional headers consume additional bandwidth.
- Poor choice of fragment sizes can greatly increase the cost of delivering a datagram. A pathological case of TCP implementation at the time sent *ten* 1024-byte TCP segments through ARPAnet which had a MTU size of 1006 bytes, resulting in 20 fragments instead of the source host sending 11 966-byte TCP segments.

Second, a single fragment loss requires the higher level protocol to retransmit all of the data in the original datagram, even if most of the fragments were received correctly. A popular at the time Proteon network interface did not have sufficient interface buffering and always dropped the second packet of two back-to-back packets, resulting in unnecessary TCP or application-level retransmissions.

The paper also discussed the difficulties in efficient reassembly, due to inadequate information about packet fragments carried in IP header (e.g., how many fragments total) and arrival delays.

B. *Recommendations*

It is a common perception that Kent and Mogul [6] recommended avoiding packet fragmentation inside the network and performing fragmentation at source hosts. In fact IPv6 [5] allows only fragmentation by source hosts.

However, a careful reading of the paper shows that the paper sorted its recommendations into three categories:

1) Recommendations not involving protocol changes;
2) Recommendations for protocol changes, and
3) Recommendations for new architectures.

The authors viewed avoiding changes as a high priority consideration. If we ignore this factor, we can also divide the recommendations into two classes:

1) avoidance of network fragmentation with host reassembly, and
2) encouragement of "transparent fragmentation", or *immediate reassembly*.[2]

Note that the first recommendation can be realized by either fragmentation only by sources or by fully implementing the second recommendation for "transparent" fragmentation. In particular, the authors stated that

> *We urge consideration of transparent fragmentation whenever possible. There is little value in the ability to send fragments of one datagram along different routes, and reassembly by gateways should not be prohibitively expensive. Main memory sizes and costs are improving so rapidly that buffer space should no longer be considered the limiting*

[1]If NDN is run over a TCP or UDP tunnel, it does not need to handle fragmentation or reassembly since the TCP/UDP packets will be fragmented and reassembled by IP over the tunnel when necessary.

[2]Historically, there were large scale subnets like ARPAnet, "transparent fragmentation" means fragmentation within a subnet. If a subnet is limited to a single-hop (e.g., WiFi, Ethernet), *immediate reassembly* means hop-by-hop fragmentation and reassembly.

resource; reassembly might actually improve the switching rates of gateways by reducing the number of individually switched fragments.

C. Current Practice vs. Recommendations

Unfortunately, Kent and Mogul recommendation of "transparent fragmentation" seems to be lost over the years. Over time, the de facto practice has evolved to limiting the size of transmission units at source hosts. In addition to performance problems pointed out in [6], one big factor calling for such practice is that today fragmented IP packets are likely to be blocked by middleboxes (firewalls and NATs).

To enable source hosts to properly size outbound IP packets, IETF has developed path MTU discovery protocols [7]. Yet practitioners today seem to primarily rely on carefully choosing packet size limit to avoid network fragmentation based on the de facto 1500 byte MTU, resulting in IP packet size "ossification", i.e., being frozen at 1500 byte limit. No one dares to use larger packet size in fear of their packets being fragmented, and eventually dropped, by routers along the path.

III. Hop-by-Hop Fragmentation and Reassembly (HBH F/R) in NDN

In this section we first elaborate why we consider HBH to be the only viable option for fragmentation and reassembly in NDN architecture. Afterwards, we discuss advantages and potential issues of application of HBH F/R in NDN.

A. Why NDN Should Have Only HBH F/R?

1) Interest packets cannot be partially processed: NDN is an architecture specifically focused on data and data retrieval. Consumers throw questions to the network for the desired data (express interests) and the network is responsible for finding and returning the requested data. These interests are evaluated by routers on the path, each deciding whether the interests can be satisfied with previously cached data, or they should be aggregated with similar interests from another consumers, or further forwarded towards the anticipated location of data or data producer. In this model, to make a decision, the router must have the full interest available, which is only possible if the interest is not fragmented. Therefore, transparent HBH fragmentation should be used for interests.

2) Full data packet is necessary to match pending interests: To deliver data packets back to the requesters, NDN relies on the state created by the forwarded interests ("pending interests"). More specifically, when a data packet is received, a router tries to match it against one or multiple pending interests. This match is based on the data name and in many cases requires other parts of the data packet. The best example here is retrieval of data packets using the implicit digest [8]. If interest includes the implicit digest, routers are required to calculate the cryptographic digest over the complete data packet in order to match pending interest to the incoming data packet. Therefore, each router along the path has to have a fully assembled data packet, before it is possible to make a complete decision about this packet. By the same token, *caches should also hold full data packets for correct matching of future interests*.

3) Producers cannot fragment or properly pre-size data packets ("Path MTU" concept does not exist in NDN): Because of in-network storage, data in NDN is no longer necessarily retrieved from the original producer. As an example, it is likely that data in popular streaming videos, which makes a big portion of today's network traffic, may come from in-network storage. Since the data packets can be cached at routers and get pulled down by any future consumers, the traditional "end-to-end path" concept no longer applies, nor does the "path MTU" concept. Therefore, producers cannot presize data packets to avoid fragmentation, as it is impossible for them to know the MTU of all the future paths their data may traverse,

NDN enables any node to communicate with any other node *directly* as soon as and as long as any physical communication channel exists in between, without a prior configuration. For example, a car on the road could retrieve a morning news video file from a combined use of ad hoc connectivity (e.g., when vehicles meet on the road or at a gas station), DTN (e.g., data carried by another car), or paid cellular channels that can be the last resort for some pieces it cannot find from other cars. Therefore, different pieces of the same video file may come from different vehicles, again making it impossible for data producers to know the proper MTU size over all links. Even if it knew, packaging data with the smallest MTU would be a poor decision as it leads to unnecessary performance hit for data delivery to end consumers whose connectivity supports larger MTU sizes.

4) What about cut-through forwarding?: To avoid the delay incurred by repeated fragmentation and reassembly of an NDN data packet over a path, one proposal is to forward each fragment as soon as it is received and reassemble the fragments at the consumer ("cut-through forwarding") [9]. This is essentially IPv4-style F/R, except that it is much more complicated to implement in NDN. Since a fundamental component of the NDN architecture is that each data packet carries a signature for verifying its authenticity, an efficient mechanism must be provided to verify the authenticity of each fragment without waiting for all the fragments to arrive. Otherwise, one cannot get both the security benefit provided by the data signature and the lower-delay benefit of cut-through forwarding. Although Ghali et. al. [9] proposed an incremental authentication mechanism, their evaluation results show that this mechanism is quite slow in software implementation. Moreover, since the matching between interests and data is primarily based on data names, every fragment (not just the first one) needs to carry the full data name. In addition, interest matching can also be based on additional fields in a data packet, which is difficult (and perhaps impossible) to handle with fragments since each fragment contains only part of the data in the original packet. Given these reasons, we prefer the simplicity of the HBH F/R approach over this option.

B. Benefits

HBH F/R not only ensures the correct delivery of interest and data packets as explained above, but also provides multiple advantages compared to alternative approaches. First, proper implementation of HBH F/R protocol (e.g., NDNLP [10]) can efficiently recover from lost or corrupted fragments locally, which reduces packet delivery failures that require application-level retransmissions and avoids the inefficiency problem in IPv4-style F/R that one missing fragment results in the unnecessary transmission of all the remaining fragments. Second, given that missing fragments can be efficiently recovered, applications can make use of larger MTU sizes enabled by advanced link-layer networking technologies, and consider their own cost tradeoff in deciding data packet size, without worrying about confining the packet size to the smallest denominator MTU along the data retrieval path.

C. Discussion

Applications may desire to send large packets, say to reduce the signing cost, but one should also keep in mind that using a large packet size can increase the store-and-forward delay across every hop, an intrinsic factor in packet-switched networks However, whether the packet size would lead to noticeable store-and-forward delay depends on the network speed. A 1MB packet crossing a 1Gbps link has a store-and-forward delay of 8msec, but this delay is reduced to 0.8msec at 10Gbps, which is negligible compared to propagation delay.

HBH F/R does not mean that all producers can just make data packets arbitrarily big. One still needs to carefully choose packet size with a well engineered value based on common MTU sizes in use, in order to avoid a packet being fragmented and reassembled multiple times along the retrieval path.

IV. Conclusion

This memo serves two objectives. First, we wanted to clarify commonly misinterpreted recommendations by Kent and Mogul [6] regarding the use of fragmentation. The reported glitches in early IP deployment (e.g., the Proteon router's inability to handle back-to-back fragments) contributed to the notion that letting network do fragmentation can only be harmful to overall performance. However, we should clearly distinguish transient implementation issues from design defects. The former should be identified and fixed, but should not be viewed as the fault of the latter.

Second and the most important, we provided reasons why NDN necessarily adopts hop-by-hop fragmentation and reassembly (HBH F/R). As an information-oriented architecture, each router in NDN network independently process request for information (interest packet) and information itself (data packet). In order to do this processing, a complete packet has to be received by the router. In addition to that, given that end-to-end and path MTU concepts no longer apply in NDN, fragmentation and segmentation cannot be pushed to the producers. In other words, data can be retrieved from any in-network cache, and data producers cannot predict MTUs for paths to all potential consumers. Finally, HBH F/R support can effectively avoid the problem of frozen packet size as we have seen with today's IP networks. The whole network no longer needs to limit its MTU size to the smallest denominator.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of CoNEXT*, 2009.

[2] L. Zhang et al., "Named data networking (NDN) project," NDN, Tech. Rep. NDN-0001, October 2010.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM Computer Communication Reviews*, July 2014.

[4] J. Postel, "Internet protocol," RFC 791, Sep. 1981.

[5] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," RFC 2460, Dec. 1998.

[6] C. A. Kent and J. C. Mogul, "Fragmentation considered harmful," *ACM SIGCOMM*, 1987.

[7] J. Mogul and S. Deering, "Path MTU discovery," RFC 1191, Nov. 1990.

[8] NDN Project, "NDN Packet Format Specification," Online: http://named-data.net/doc/ndn-tlv/, 2014.

[9] C. Ghali, A. Narayanan, D. Oran, and G. Tsudik, "Secure fragmentation for content-centric networks," Online: http://arxiv.org/abs/1405.2861, May 2014.

[10] J. Shi and B. Zhang, "NDNLP: A link protocol for NDN," NDN, Tech. Rep. NDN-0006, 2012.