CCNx End-To-End Fragmentation draft-mosko-icnrg-ccnxfragmentation-01

Abstract

This document specifies an end to-end fragmentation protocol for breaking a Content Object into several smaller pieces in order to fit within a network MTU. The fragmentation protocol does not provide a secure binding of the fragments to the original object. This is left to the receiving endpoint. Midpoints may only serve fragments from their cache if they have assembled and verified the complete Content Object.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”

This Internet-Draft will expire on September 7, 2015.

Copyright Notice

Table of Contents

TOC

1. Introduction

This document specifies an end-to-end fragmentation protocol for breaking a Content Object into several smaller pieces in order to fit within a network MTU. The fragmentation protocol does not provide a secure binding of the fragments to the original object. This is left to the receiving endpoint. Midpoints may only serve fragments from cache if they have assembled and verified the complete Content Object.

Packets are represented as 32-bit wide words using ASCII art. Because of the TLV encoding and optional fields or sizes, there is no concise way to represent all possibilities. We use the convention that ASCII art fields enclosed by vertical bars "|" represent exact bit widths. Fields with a forward slash "/" are variable bitwidths, which we typically pad out to word alignment for picture readability.

TODO -- we have not adopted the Requirements Language yet.

TOC

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.) [RFC2119].

TOC

2. Protocol Description

The principle of operation for fragmentation in CCNx 1.0 is that intermediate systems should not have to fragment packets. An Interest message is always fragmented to the minimum MTU size and the forward path's MTU size is recorded in the Interest message so that a system sending back a Content Object can fragment it

to the the correct size for the path. An intermediate system's Content Store may store only pre-fragmented objects and respond only if those fragments satisfy the requirements of an Interest's MTU, otherwise it will be considered a cache miss.

Because the Interest is the path discovery mechanism for Content Objects, all Interests MUST carry an InterestFragmentationHeader that includes information about the forward path's MTU so the reverse path MTU may be known at the node responding with a Content Object.

The minimum MTU required is 1280 octets. Any system implementing a physical layer with a smaller MTU must implement link fragmentation, for example using a PPP layer over the small MTU network.

Systems MUST create fragment streams in the most compressed packing. That is, all fragments except the last MUST be fragmented to the same size. This means that all Interests are fragmented to 1280 bytes except the last fragment. A Content Object may be fragmented to any size no more than the path MTU discovered, but all fragments except the last MUST be the same size. This ensures that any two fragment streams of the same Content Object with the same MTU have the same representation.

When an end system creates a fragment stream, it generates a 64-bit number for the FragmentStreamID. This number identifies a contiguous stream of fragments and allows an end system to use the FragmentStreamID for reassembly. An intermediate system uses the FragmentStreamID of a Content Object to ensure that only one stream of Content Object fragments follow a reverse PIT entry.

A system SHOULD use a random number for an Interest's FragmentStreamID. This avoids easy denial-of-service attacks by replying with junk for known fragment stream IDs. A fragmented Content Object carries both its own FragmentStreamID, which SHOULD be based on the ContentObjectHash, and the corresponding Interest FragmentStreamID to facilitate matching on the reverse PIT path.

If the Maximum Path MTU of a Content Object fragment is larger than the supported MTU on an egress interface, the fragment stream should be dropped on that interface, even if some of the fragments fit within the MTU.

Fragments are identified by a serial counter FragNum, which ranges from 0 - 63. Forwarders and end systems should drop duplicate fragments, identified by the tuple {FragmentID, FragNum}.

If all fragments are not received within a system-dependent timeout, a system re-assembling fragments should timeout. If the re-assembly of an Interest times out before the PIT entry, the PIT entry on the local system should be removed to allow a new fragment stream to arrive. If the re-assembly of a Content Object times out, the received fragments bitmask of the PIT should be cleared to allow a new stream of Content Objects to arrive.

TOC

2.1. Interest Fragmentation

If an Interest does not fit with 1280 bytes, then it must be fragmented to fit within 1280 bytes. There is no path MTU discovery for Interests.

As an Interest goes through the FIBs, it records the minimum path MTU based on the egress interface's MTU. A Content Object sent in response must be fragmented to less than or equal to the minimum path MTU. A forwarder may choose to put 1280 in the Minimum Path MTU field even if it supports larger MTUs.

Interests follow the FIB and all fragments of an Interest (i.e. the same fragment id) should follow the same FIB choices. If at a later time a similar interest arrives with a smaller minimum path MTU, it should be forwarded even though it is similar, to ensure that a returned Content Object is fragmented to a size that satisfies the Interest's path.

A forwarding node must examine the Interest name to determine its forwarding. This requires that the forwarding node re-assemble the front of the Interest to examine the name. In a typical case, this means that the node must receive fragment 0 to have enough prefix name components to compute the route. A system MAY discard out-of-order fragments after fragment 0 during this re-assembly, and once fragment 0 arrives and the system constructs a PIT entry with the routing, it should send a control message along the Fragment Stream ID's reverse path to cause the source to resend the interest stream, which can now be forwarded out of order. Or, it may buffer out-of-order fragments.

A system that receives an Interest encapsulated in a packet larger than 1280 octets must discard it.

TOC

2.2. Content Object Fragmentation

When forwarding a Content Object along the reverse path of the PIT, a fragment stream may only be forwarded along reverse PIT entries for which it satisfies the reverse path minimum MTU.

A PIT entry should only be removed once all fragments of a fragment stream pass through, or it times out. Because the FragCnt is limited to 63, a system may match a first stream's Fragment ID and use a single 64-bit mask.

A Content Object is fragmented based on the Interest minimum path MTU. It carries an "Maximum Fragment MTU" field set to the maximum fragment size, which must be no more than an Interest's minimum path MTU. Because a fragment stream may only satisfy PIT entries with larger or equal minimum path MTU, all fragments must carry the Object's fragmentation size. An intermediate node may, for example, receive the last fragment first, so even if fragments were packed to maximum size, the forwarder could not infer which PIT entries the object satisfies without know the fragment stream's fragmentation size

---

TOC

3. Packet Formats

End-to-end fragmentation uses a hop-by-hop TLV header for fragmentation. There is one header for Interests and one header for Content Objects.

Type	Abbrev	Name	Description
%x0006	T_INTFRAG	Interest fragmentation header (Interest Header)	Hop-by-hop fragmentation header for Interests.
%x0007	T_OBJFRAG	Content Object fragmentation header (Content Object Header)	Hop-by-hop fragmentation header for Content Objects.

Table 1: Hop-by-hop Header Types

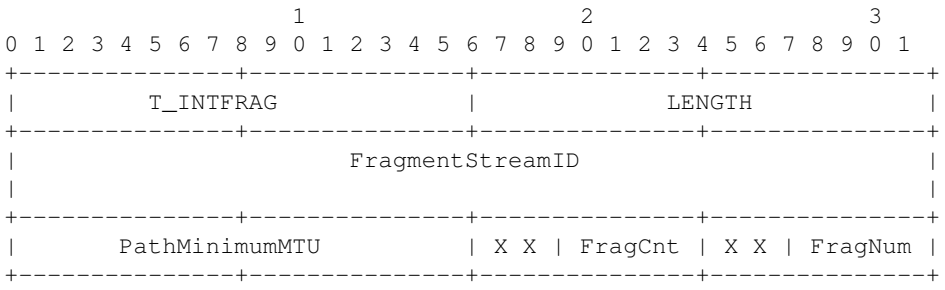
Both fragmented Interests and Content Objects use the fields FragCnt and FragNum. The count is the number of fragments in the message, which has a minimum of 1. FragNum is the 0-based fragment number. Sequential fragment numbers represent sequential byte boundaries of the original object.

TOC

3.1. Interest Header

The field FragmentStreamID identifies a contiguous stream of fragments. It SHOULD be a random number.

The field PathMinimumMTU is updated per-hop to measure the minimum path MTU of the interest's reverse path.



TOC

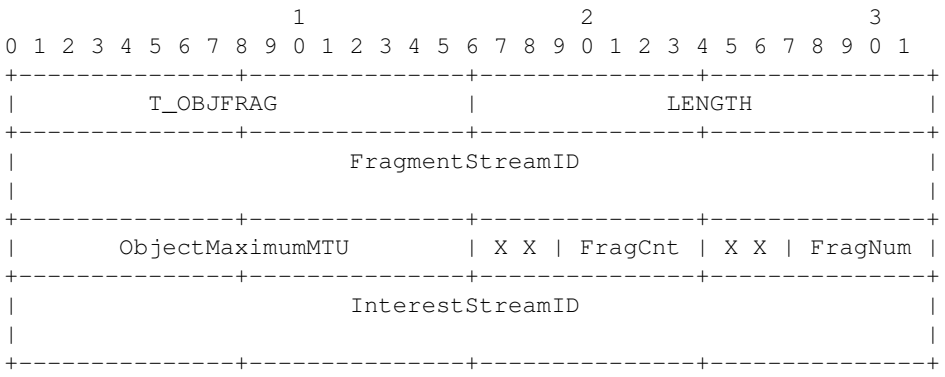
3.2. Content Object Header

The field FragmentStreamID identifies a contiguous stream of fragments for the Content Object. It SHOULD be derived from the Content Object's hash, so two objects with the same Fragment Stream ID represent the same fragmented object.

The field InterestStreamID is the Fragment ID of the corresponding Interest that the object is answering. This allows PIT matching without having to reconstruct the Content Object. It makes Content Objects specific to a given Interest similarity hash.

The field ObjectMaximumMTU is the maximum size of any fragment in the fragment stream. This allows a forwarder to match a content object fragment stream against a reverse path MTU size and not send a fragment

stream that will not fit down a path.



TOC

4. Cache Considerations

Objects should be reassembled before sending from cache, to ensure all fragments exist at the cache.

Single fragment Interests may be satisfied from cache. A system may choose to reassemble Interests to try and answer from cache. If a cache miss, the original fragment stream should be forwarded.

TOC

5. Acknowledgements

TOC

6. IANA Considerations

TODO: Work with IANA to define the type space for: Hop-by-hop header types.

All drafts are required to have an IANA considerations section (see Guidelines for Writing an IANA Considerations Section in RFCs (Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” May 2008.) [RFC5226] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

TOC

7. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 (Rescorla, E. and B. Korver, “Guidelines for Writing RFC Text on Security Considerations,” July 2003.) [RFC3552] for a guide.

TOC

8. References

TOC

8.1. Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

TOC

8.2. Informative References

[CCN] PARC, Inc., “CCNx Open Source,” 2007.

[RFC3552] Rescorla, E. and B. Korver, “Guidelines for Writing RFC Text on Security Considerations,” BCP 72, RFC 3552, July 2003 (TXT).

[RFC5226] Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” BCP 26, RFC 5226, May 2008 (TXT).

TOC

Author's Address

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA
Phone: +01 650-812-4405
Email: marc.mosko@parc.com