ICNRG
Internet-Draft
Intended status: Experimental
Expires: December 3, 2016

M. Mosko
PARC, Inc.

June 1, 2016

CCNx Content Object Chunking
draft-mosko-icnrg-ccnxchunking-02

Abstract

This document specifies a chunking protocol for dividing a user payload into CCNx Content Objects. This includes specification for the naming convention to use for the chunked payload and a field added to a Content Object to represent the last chunk of an 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 December 3, 2016.

Table of Contents

1. Introduction 3
1.1. Requirements Language 3
2. Chunking 4
2.1. Cryptographic material 5
2.2. Examples 5
3. TLV Types 6
3.1. Name Types 6
3.1.1. Chunk Number 6
3.2. Protocol Information 6
3.2.1. EndChunkNumber 7
4. Acknowledgements 8
5. IANA Considerations 9
6. Security Considerations 10
7. References 11
7.1. Normative References 11
7.2. Informative References 11
Author's Address 12

1. Introduction

CCNx Content Objects [CCNSemantics] are sized to amortize cryptographic operations over user data while simultaneously staying a reasonable size for transport over today's networks. This means a Content Object is usually within common UDP or jumbo Ethernet size. If a publisher has a larger amount of data to associate with a single Name, the data should be chunked with this chunking protocol. This protocol uses state in the Name and in an optional field within the Content Object. A chunked object may also have an external metadata content object that describes the original pre-chunked object.

CCNx uses two types of messages: Interests and Content Objects [CCNSemantics]. An Interest carries the hierarchically structured variable-length identifier (HSVLI), or Name, of a Content Object and serves as a request for that object. If a network element sees multiple Interests for the same name, it may aggregate those Interests. A network element along the path of the Interest with a matching Content Object may return that object, satisfying the Interest. The Content Object follows the reverse path of the Interest to the origin(s) of the Interest. A Content Object contains the Name, the object's Payload, and the cryptographic information used to bind the Name to the payload.

This specification adds a new segment to the Name TLV for conveying the chunk number. It updates [CCNMessages]. It also provides guidelines for the usage of the Key Locator in chunked objects.

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.

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 [RFC2119].

2. Chunking

Chunking, as used in this specification, means serializing user data into one or more chunks, each encapsulated in a CCNx Content Object. A chunk is a contiguous byte range within the user data. One segment in the Name of that Content Object represents the chunk number. A field in the Content Object - only mandatory in the final chunk - represents the end of the stream. Chunks are denoted by a serial counter, beginning at 0 and incrementing by 1 for each contiguous chunk. The chunking ends at the final chunk. No valid user data exists beyond the final chunk, and reading beyond the final chunk MUST NOT return any user data.

Chunking MUST use a fixed block size, where only the final chunk MAY use a smaller block size. This is required to allow a reader to seek to a specific byte offset once it knows the block size. The blocksize may be inferred from the size of the first chunk of user data. The first chunk of user data may not be chunk 0.

Because of the requirement for a fixed block size, the inclusion of certain cryptographic fields in the same content objects as user data would throw off the ability to seek. Therefore, it is RECOMMENDED that all required cryptographic data, such as public keys or key name links, be included in the leading chunks before the first byte of user data. User data SHOULD then run continuously and with the same block size through the remainder of the content objects.

This draft introduces a new Name path segment TLV type, called the ChunkNumber name segment. The ChunkNumber name segment is the serial order of the chunks. It MUST begin at 0 and MUST be incremented by 1. The ChunkNumber name segment is appended to the base name of the user data, and is usually the last name segment.

The new Content Object field is the EndChunkNumber. It MUST be included in the Content Object which is the last chunk of user data, but SHOULD be present at the earliest time it is known. The value of the EndChunkNumber should be the network byte order value of the last ChunkNumber. For example, if 3000 bytes of user data is split with a 1200 byte block size, there will be 3 chunks: 0, 1, and 2. The EndChunkNumber is 2.

The EndChunkNumber may be updated in later Chunks to a larger value, as long as it has not yet reached the end. The EndChunkNumber SHOULD NOT decrease. If a publisher wishes to close a stream before reaching the End Chunk, it should publish empty Content Objects to fill out to the maximum EndChunkNumber ever published. These padding chunks MUST contain the true EndChunkNumber.

2.1. Cryptographic material

Chunk 0 SHOULD include the public key or key name link used to verify the chunked data. It is RECOMMENDED to use the same key for the whole set of chunked data. If a publisher uses multiple keys, then the public key or key name link for all keys SHOULD be in the leading chunks before any user data.

The rationel for putting all cryptographic data up front is because the protocol requires using a fixed block size for all user data to enable seeking in the chunked stream.

2.2. Examples

Here are some examples of chunked Names using the Labeled Content Identifier URI scheme in human readable form (ccnx:).

In this example, the content producer publishes a JPG that takes 4 Chunks. The EndChunkNumber is missing in the first content object (Chunk 0), but is known and included when Chunk 1 is published. It is omitted in Chunk 2, then appears in Chunk 3, where it is mandatory.

```
ccnx:/Name=parc/Name=picture.jpg/Chunk=0  --
ccnx:/Name=parc/Name=picture.jpg/Chunk=1  EndChunkNumber=3
ccnx:/Name=parc/Name=picture.jpg/Chunk=2  --
ccnx:/Name=parc/Name=picture.jpg/Chunk=3  EndChunkNumber=3
```

In this example, the publisher is writing an audio stream that ends before expected so the publisher fills empty Content Objects out to the maximum ChunkNumber, stating the correct EndChunkNumber. Chunks 4, 5, and 6 do not contain any new user data.

```
ccnx:/Name=parc/Name=talk.wav/Chunk=0  --
ccnx:/Name=parc/Name=talk.wav/Chunk=1  EndChunkNumber=6
ccnx:/Name=parc/Name=talk.wav/Chunk=2  --
ccnx:/Name=parc/Name=talk.wav/Chunk=3  EndChunkNumber=3
ccnx:/Name=parc/Name=talk.wav/Chunk=4  EndChunkNumber=3
ccnx:/Name=parc/Name=talk.wav/Chunk=5  EndChunkNumber=3
ccnx:/Name=parc/Name=talk.wav/Chunk=6  EndChunkNumber=3
```

3. TLV Types

This section specifies the TLV types used by CCNx chunking.

3.1. Name Types

CCNx chunking uses two new Name types: Chunk Number and Chunk Metadata.

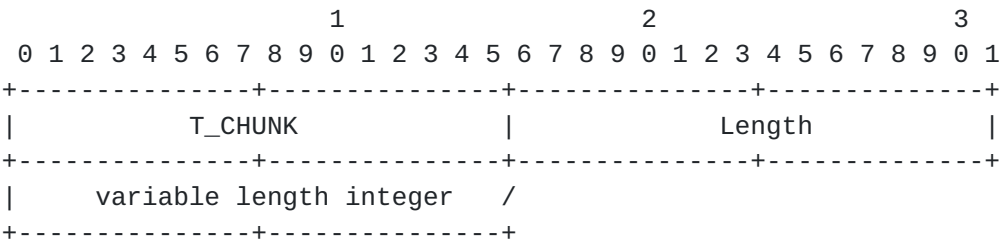
Type	Abbrev	Name	Description
%x0010	T_CHUNK	Chunk Number (Section 3.1.1)	The current Chunk Number, is an unsigned integer in network byte order without leading zeros. The value of zero is represented as the single byte %x00.

Table 1: Name Types

3.1.1. Chunk Number

The current chunk number, as an unsigned integer in network byte order without leading zeros. The value of zero is represented as the single byte %x00.

In ccnx: URI form, it is denoted as "Chunk".



3.2. Protocol Information

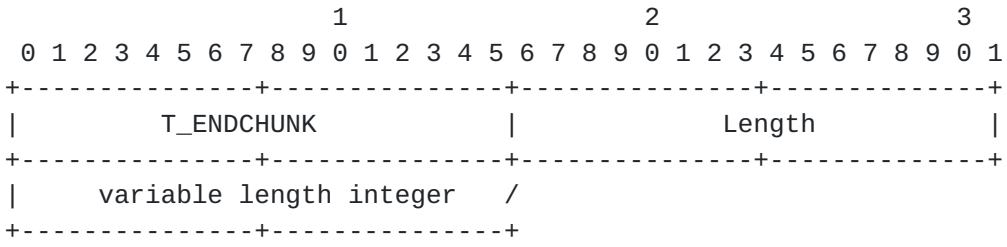
CCNx chunking introduces one new TLV for use in a Content Object.

Type	Abbrev	Name	Description
%x000C	T_ENDCHUNK	EndChunkNumber (Section 3.1.1)	The last Chunk number, as an unsigned integer in network byte order without leading zeros. The value of zero is represented as the single byte %x00.

Table 2: Content Object Types

3.2.1. EndChunkNumber

The ending chunk number, as an unsigned integer in network byte order without leading zeros. The value of zero is represented as the single byte %x00.



4. Acknowledgements

5. IANA Considerations

The draft adds new types to the CCNx Name Segment Types registry and the CCNx Content Object Types registry.

6. Security Considerations

This draft does not put any requirements on how chunked data is signed or validated.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[CCNMessages] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format (Internet draft)", 2016, <http://tools.ietf.org/html/draft-irtf-icnrg-ccnxmessages-02>.

[CCNSemantics] Mosko, M., Solis, I., and C. Wood, "CCNx Semantics (Internet draft)", 2016, <http://tools.ietf.org/html/draft-mosko-icnrg-ccnxsemantics-03>.

[CCNx] PARC, Inc., "CCNx Open Source", 2007, <http://www.ccnx.org>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

Author's Address

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com