CCNx chunking protocol

draft-mosko-icnrg-ccnxchunking-03

IETF 121, Dublin

https://github.com/mmosko/draft-irtf-icnrg-chunking

Marc Mosko

Hitoshi Asaeda

NICT

Outline

- A brief history of chunking / segmentation
- Some examples
- The proposal

Brief History

- The original CCNx 0.x protocol used a segmentation convention to split a user object into multiple content objects.
  - A content object was called a “segment.”
  - There was a field called “FinalBlockId” independent of the %00 and %FD command markers.
  - You could see an email I sent to the NDN list about the 0.8 segmentation [1].
- In 2015-2016, we wrote the first draft-mosko-icnrg-ccnxchunking-{00,01,02}. At the time, it was not pursued as an RG document.
- We chose to rename segmentation into chunking to clearly break from the previous usages.

[1] https://www.lists.cs.ucla.edu/pipermail/ndn-interest/2018-June/002208.html

Usage

- Chunking takes a single object and chunks it into contiguous byte Content Object payloads.
- It adds a Chunk ID sequence number to the name for each chunk.
- It adds a FinalChunkId field to the content object with the last chunk. When the value of the field matches the name component value, it is the last chunk.

Examples (1)

Name = ccnx:/foo/bar.jpg/ChunkId=0
Payload = <500 bytes>
FinalChunkID = 0
Validation Alg = {key locator}
Validation Payload = <bytes>

A single-chunk object.

Examples (2)

Name = ccnx:/foo/bar.jpg/ChunkId=0
Payload = <1100 bytes>
FinalChunkID = 2
Validation Alg = {key locator}
Validation Payload = <bytes>

Name = ccnx:/foo/bar.jpg/ChunkId=1
Payload = <1100 bytes>
FinalChunkID = 2
Validation Alg = {key locator}
Validation Payload = <bytes>

Name = ccnx:/foo/bar.jpg/ChunkId=2
Payload = <600 bytes>
FinalChunkID = 2
Validation Alg = {key locator}
Validation Payload = <bytes>

A 3-chunk object.

Examples (3)

Name = ccnx:/foo/cat.jpg/ChunkId=0
FinalChunkID = 3
Validation Alg = {key locator or cert}
Validation Payload = <bytes>

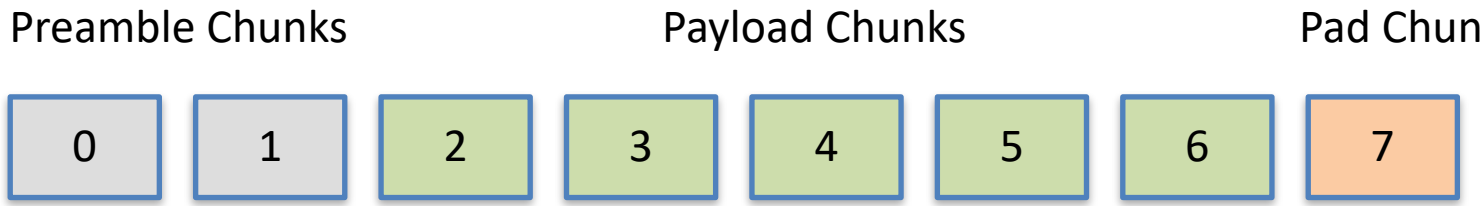
A 3-chunk object that pre-loads crypto data and then maintains equal chunk sizes.

Name = ccnx:/foo/cat.jpg/ChunkId=1
Payload = <1300 bytes>
FinalChunkID = 3
Validation Alg = {key id}
Validation Payload = <bytes>

Name = ccnx:/foo/cat.jpg/ChunkId=3
Payload = <900 bytes>
FinalChunkID = 3
Validation Alg = {key id}
Validation Payload = <bytes>

Name = ccnx:/foo/cat.jpg/ChunkId=2
Payload = <1300 bytes>
FinalChunkID = 3
Validation Alg = {key id}
Validation Payload = <bytes>

Summary



Allow for early close to unexpected end of stream.

The Rules

- Every chunk MUST have a ChunkNumber name segment, beginning at 0 and incrementing by 1.
- The leading chunks MAY have missing or empty Payload TLVs and convey only cryptographic [or other] information.
- All Content objects with a Payload MUST use the same block size except the last. They MUST be ChunkNumber contiguous.
- The last chunk MUST have an EndChunkNumber TLV and the value MUST be equal to the ChunkNumber TLV value.
- Content Objects before the last chunk MAY have an EndChunkNumber TLV with the expected last chunk number. These hints MAY be updated in subsequent Content Objects but SHOULD NOT decrease.
- If the final chunk has a ChunkNumber less than a previously published EndChunkNumber, the publisher SHOULD pad out the chunks with empty Content Objects that have the true EndChunkNumber.

What I would change

- The requirement for equal-sized payloads until the last chunk is overly restrictive. I would make equal-sized chunks (for seeking) optional with a “ChunkSize” field in ContentObject chunk 0. Otherwise, the Payload can be any size in each chunk.

Why not use FLIC?

- Yes, you should. This allows offloading all cryptographic operations to the root manifest, so you do not need ValidationArgs or ValidationPayload anywhere else.
- The SegmentedSchema name constructor uses Chunked named to ensure each name is unique. This uses the Chunking convention. [obviously the name should be updated]
- Removing the equal-sized chunks is good for FLIC, as it allows the encoder to vary sizes as it wants.

Conclusion

- We wish to adopt this as an RG document.
- The only proposed changed from -03 is to remove the fixed-sized payloads and make it optional if a field is present in Chunk 0.

Q&A