File-Like ICN Collections (FLIC)

draft-irtf-icnrg-flic-05

IETF 118, Prague

Marc Mosko

SRI/PARC

Dave Oran

Network Systems Research & Design

Outline

- What FLIC does (super quick recap)
- Updates since -04
- Next steps

What FLIC does

- It provides a manifest of hashes that make up all the segments of a piece of application data.
- The manifest is hierarchical – that is the hash pointers can point to application data or to more manifests.
- There is a canonical traversal order. Metadata could provide other traversal hints, such as for video.
- FLIC has its own, extensible, encryption mechanism. Manifest encryption does not need to be related to content encryption.
- FLIC has several Interest construction techniques. The publisher can choose one or more of these naming techniques. More techniques could be added.

Main Update

- Segmented Schema
  - A schema defines how a consumer constructs an Interest name from the manifest entries.
  - Segmented Schema means to use a name prefix plus a segment number plus a hash.
  - A schema applies to a single Name Constructor.
    - Typically one name constructor is used for Manifest objects and one is used for app data objects.

Main Update

- Segmented Name Schema
  - The publisher uses segment # in the name, e.g. /foo/bar/1, /foo/bar/2
  - For each Name Constructor that uses Segmented Name, the consumer must track the segment number.
  - For sane use, the segment numbers should track the Manifest in-order traversal, as the App data is defined to be reconstructed by that traversal.

How to track Segment

- Option 1 (the previous default)
  - The consumer stars with the first hash pointer of the Name Constructor and assigns it 0, then it must go in-order through the manifest and increment the segment id.
  - The consumer must remember the number between manifest objects.
  - The consumer must retrieve every manifest object in-order.

How to track Segment # (part 2)

- Option 2 (the previous alternative)
  - The publisher uses Annotated Hash Pointers and has an annotation that explicitly gives the segment #.
  - For sane operation, the segment numbers should go in traversal order.
  - Some applications, like audio/video media, might want to skip to a different segment number, or data de-dup apps might refer to common byte strings.

How to track Segment # (part 3)

- Option 3 (the new piece)
  - Each Hash Group in a Manifest object includes metadata that says what the starting Segment ID is. A consumer then only needs to know its offset within that one object to create the Segment Id.

Notes on Segment Numbers (1)

- What FLIC requires
  - For a given name prefix, a segment number is only used once (i.e. each segment number has a single unique object hash) in the data names.
  - The consumer is not required to enforce this.
- What FLIC allows
  - A segment number may be used more than once.
  - A segment number may not be used by the publisher.
  - A consumer may skip segments or go out-of-order.

Notes on Segment Numbers (2)

- For general sanity:
  - The publisher should use 0, …, N-1 as in-order segment numbers when creating the data objects.
  - The manifest should use a single mechanism (annotated pointers or StartSegmentIds).
  - The in-order traversal should fetch 0, …, N-1 in order.
- None of those are mandatory

Complications (1)

- Q: What happens if a publisher uses an Annotated Hash Group but only includes a Segment Id annotation for some pointers?
- A: The publisher must include a StartSegmentId in the Hash group and the consumer proceeds as Option 3. If a pointer has an explicit SegmentId, the consumer uses that and does not increment the implicit segment id.

Complications (2)

- Q2: What if the Hash Group uses Segmented Naming, but the Hash Group does not have a StrartSegmentId?
- A2: Then it must be an annotated hash group and every pointer must have an explicit segment id. Otherwise, it is a malformed manifest and should be discarded.

Complications (3)

- Q3: A publisher uses a regular Hash Group with Segmented Name schema. It includes a StartSegmentId for each Hash Group. But the numbers overlap.
  - E.g. Manifest #1 has StartSegmentId 0 and 10 elements and Manfest #2 has StartSegmentId 5 and 10 elements.
- A3a: The app data has 20 elements, with the 5-9 and 10-14 the same.
- A3b: Don’t do this.

Conclusion

- The data associated with a Name Constructor is assembled according to the Manifest in-order traversal.
  - Segment numbers do not guide the re-assembly. They are just part of the name of the pieces.
  - A publisher can use segment numbers in non-sane ways, but if the in-order traversal is correct, then that use, although screwy, is correct. We don't recommend this, but we do not prohibit it.
  - Data de-duplication is the only valid use case I can think of.

Q&A