Nameless Objects

Marc Mosko[1]*

Abstract
In CCN, the Name of an Interest message determines where the request is routed. It also determines which Content Objects match that Interest. If one wishes to place Content Objects along different routing paths, then one must either name them differently, use Links as a means of indirection, or use Encapsulation to "tunnel" the objects under a different Name. Using Nameless objects solves the problem in a different way that allows graceful migration of data between replicas without resigning or renaming.

Keywords
Content Centric Networks

[1] *Palo Alto Research Center*
*__Corresponding author__: marc.mosko@parc.com

Contents

Introduction

In CCNx 1.0, an Interest message carries a Name and an optional ContentObjectHash. A Content Object matches an Interest if the names are exactly equal. If the Interest also specified the ContentObjectHash restriction, then the SHA-256 hash of the entire Content Object must be computed and matched against the restriction in the Interest.

In CCNx, the Name is also used for routing. Therefore, the name not only identifies the piece of content, but also indicates where the content lives, or at least the name of the authority to say where it lives. The authority could respond with a Link to a different location, such as a different replica. Because the link it signed by the authority, the client may trust the transfer of security context. The link could be to just a name, but we believe for a transfer of security context the link should enumerate the (Name, KeyIdRestriction, ContentObjectHashRestriction) tuple.

There are several problems with using a Link redirect. First, the authority must be aware of the replica. Second, because the name is included in the ContentObjectHash, the ContentObjectHash implies a specific name. To move a content object to a new replica under new routing means that the ContentObject name must change and thus the ContentObjectHash will change too. Because of this, the authority must issue either a link for only (Name, ContentObjectHash), or the new replica must re-sign the content with its key and the authority issue a link for (Name, KeyId, ContentObjectHash), or the authority must re-sign the content and send it to the replica via tunneling.

All of the above make it difficult in CCNx to move a piece of content to a different replica and have it served under different routing names. Therefore, we introduce Nameless Objects as a way to solve the locator/identifier separation in CCNx.

1. Nameless Objects

We introduce the concept of "Nameless Objects." These are a Content Object without a Name. They can only be addressed by the ContentObjectHash self-certified name. An Interest would still have a Name, which is used for routing, and a ContentObjectHash, which is used for matching. On the reverse path, however, if the Content Object's name is missing, it is a "Nameless Object" and only matches against the ContentObjectHash.

This means that a requester can fetch the data from anywhere it lives.

For example, a user could issue an Interest for `/parc/csl/slides.pdf` and receive back a Catalog[1] and Manifest that indicates there are three replicas — `/akamai`, `/xerox/cloud`, and `/ccnxcdn` – that all serve the content. The manifest would then enumerate the Content Object hash values for each chunk of the slides. The client would then issue an Interest to its preferred replica, say `/xerox/cloud` using the list of Content Object hashes. The returned content objects would be Nameless Objects that match the SHA-256 hashes.

Nameless Objects are also useful even without replica redirection. For example, a DNA sequence is encoded by gene with a root manifest per gene. The encoding is a root manifest with a name (e.g. /hgvs.org/foo) that then points to a tree of Nameless manifests. One could retrieve a gene sequence by that canonical name. A first researcher organizes them

[1] The topic of a Catalog and enumerating replicas is beyond the scope of this paper, it is only used as a motivating example

in named parent manifest (e.g. /ucsc.edu/seq1) for a specific purpose, pointing only to the Nameless manifest tree of each sequence. A second researcher could organize the same genes in to a different sequence (e.g. /berkeley.edu/seq2) using a different named parent manifest using the same nameless manifest tree for each gene. Thus, someone downloading both sequences only retrieves the unique manifests created by each researcher, the unique genes of each sequence, and the shared genes. This has realized a form of data de-duplication.

Because the client trusted the initial Manifest response, which is a named object with proper cryptographic signature, it will trust the Manifest and the enumerated Content Object hashes.

Note that the Nameless Objects are truly a placeless object. They have no name, so no implied routing.

Having no name means that they will never match an entry in the PIT that requested something only by Name, or perhaps Name and KeyId. That is the desired action, because a retrieval by hash does not obey routing, so it could be a so-called "off-path attack". For example, if a Content Object could be retrieved by only Contnet Object Hash, but also carried a name, an attacker, Eve, could issue an Interest with the name, for example, of `/hacker/attack` with a Content Object hash, and have it return a content object with the name `/parc/csl/slides.pdf`, even though the Content Object is not those slides and the Interest would never have been routed to PARC. This could cause a timing attack against valid requests.

To publish a Nameless Content Object, one would first create a signed Manifest with an authoritative name in it. The Manifest would need to enumerate the possible content distribution names and the Nameless object's Content Object hashes. When the storage replicas change, the manifest must be changed, which could be expensive for a large manifest.

As an alternative, one could publish both the Manifest and the Content Objects as nameless objects. Then, one would publish a single small Manifest that only links to the Nameless Manifest via the available replicas. This extra step of indirection could by more flexibility in publishing.

The notional PIT table normally describes indexing Interests by the tokens (Name), (Name, KeyId), and (Name, KeyId, ContentObjectHash). Using Nameless Objects means that an additional index by the token (ContentObjectHash) alone is necessary. This new index should contain backpointers to all the (Name, KeyId, ContentObjectHash) entries of the same ContentObjectHash to facilitate satisfying those entries. This description of the notional PIT table is only for illustration of the correct behavior (described in outline below) and any given implementation may vary the table organization.

Similarly, the Content Store (CS) may enable lookup by only (ContentObjectHash).

PIT table aggregation is unchanged, as

In detail, the forwarding rules are:

1. Receive Interest
   (a) The Interest must have a name.
   (b) Create PIT entry.
      i. If Interest has a ContentObjectHashRestriction, must be able to match PIT entry without a Name.
   (c) Aggregation rules apply as normal and now include additional keys by (KeyIdRestriction, ContetObjectHashRestriction) and (ContentObjectHashRestriction).
   (d) Apply normal forwarding rules.
2. Receive ContentObject
   (a) ContentObject has a Name
      i. Apply normal forwarding rules.
   (b) Otherwise
      i. Satisfy all PIT entry on (Name, ContetObjectHashRestriction) that matches on (ContetObjectHash) alone.
      ii. Satisfy all PIT entry on (Name, KeyId, ContetObjectHashRestriction) that matches on (KeyId, ContetObjectHash).

2. Conclusion

Nameless objects provide a means to move Content between storage replicas without having to rename or re-sign the content objects for the new name. In one case, the entire set of content objects and manifest can be nameless objects so only a single, small object needs to enumerate the storage replicas and be cryptographically signed.

References