NDN, Technical Report NDN-0025, 2014. http://named-data.net/techreports.html
Revision 1: September 25, 2014

Fetching content in Named Data Networking with embedded manifests

Ilya Moiseenko
UCLA
iliamo@ucla.edu

INTRODUCTION

Manifests are proposed to be a special type of content in Named Data Networking that contains meta-information about other Data packets: a sequence of Data segments or completely independent information objects. While a great variety of useful meta-information exists, this document focuses on the case when manifest contains a list of Data packet names.

For example, a manifest containing full names (prefix + digest of the packet) can be used by the consumer application for faster verification of data packets. Only the manifest object must be verified using the public key cryptography, whereas all other Data packets listed in the manifest can be verified by simple computation of the digest and comparison to the digest specified in already verified manifest.

An example of the contents of the manifest:
/a/b/digest01234567
/c/z/digest09876543
….

Such manifest must be fetched separately from other data packets, because it includes the digests that must be computed separately as a prior step. In a general case, manifest fetching takes an additional round-trip.

The purpose of this technical memo is to introduce the use of manifests for faster signing and verification of Data packets without requiring an additional round-trip delay for manifest fetching.

SEGMENTATION PROTOCOL

Segmentation is the operation applied to the content when it cannot fit in a single Data packet. Segmentation protocol defines how the content is split among multiple Data packets and how these Data packets are named. A straightforward way to name data segments is to use sequentially incremented segment number:
/a/b/0
/a/b/1
/a/b/2
…

Because each data segment must carry a signature, segmentation can become computationally expensive if public key cryptography is being used for every segment. At the same time, consumer applications can experience difficulties with verifying per-segment signatures using public key cryptography.

With manifests, it is possible to replace signing of data segments with:
(1) computing per-segment digest (hash),
(2) saving segment names (with digest) in the manifest, and
(3) signing the manifest using public key cryptography.

The manifest will contain names of data segments including the digest as a name component [1].
/a/b/1/digest1234
/a/b/2/digest5678
/a/b/3/digest9012
…

Manifest segment has the same name prefix as the other data segments and has a sequence number preceding the first segment number of the current batch of data segments.

An example of content segmentation with embedded manifests:
/a/b/0 - manifest segment containing /a/b/1, /a/b/2, a/b/3 names with digests
/a/b/1 – data segment
/a/b/2 – data segment
/a/b/3 – data segment
/a/b/4 – manifest segment containing /a/b/5, /a/b/6, a/b/7 names with digests
/a/b/5 – data segment
/a/b/6 – data segment
/a/b/7 – data segment
…

Allowed maximum data packet size has a direct influence on the number of names each manifest segment can contain. A single manifest segment can include hundreds of names if data packet size is large (32KB), and just a few dozens if the data packet size is small (4KB). This trade-off has an impact on the speed of signing and verification.

INTEREST SEQUENCING PROTOCOL

Large content cannot be retrieved with a single Interest packet, because it is split into multiple Data segments. In order to fetch all segments, consumer application must send a sequence of Interest packets for these Data segments.

A typical Interest sequencing protocol consists of flow and congestion control algorithm that defines Interest window size – a number of transmitted Interests waiting for corresponding Data packets.

Since manifests are embedded in the sequence of Data segments, Interest sequencing protocol does not need to perform a stand-alone round-trip fetching of the manifests.

Consumer application can verify every Data segment by calculating its digest and comparing it to the digest specified in the manifest. This operation is faster than using public key crypto for every segment.

REFERENCES

[1] Baugher, Mark, et al. "Self-verifying names for read-only named data." INFOCOM Workshops. Vol. 12. 2012.