

ICNRG
Internet-Draft
Intended status: Experimental
Expires: 25 July 2022

C. Gundogan
TC. Schmidt
HAW Hamburg
D. Oran
Network Systems Research and Design
M. Waehlich
link-lab & FU Berlin
21 January 2022

Alternative Delta Time Encoding for CCNx Using Compact Floating-Point
Arithmetic
draft-gundogan-icnrg-ccnx-timetlv-05

Abstract

CCNx utilizes delta time for a number of functions. When using CCNx in environments with constrained nodes and/or bandwidth constrained networks, it is valuable to have a compressed representation of delta time. In order to do so, either accuracy or dynamic range has to be sacrificed. Since the current uses of delta time do not require both simultaneously, one can consider a logarithmic encoding such as that specified in [IEEE.754.2019]. This document updates _CCNx messages in TLV Format_ (RFC8609) to specify this alternative encoding.

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 <https://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 25 July 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Terminology	2
3. Usage of Time Values in CCNx	3
3.1. Relative Time in CCNx	3
3.2. Absolute Time in CCNx	3
4. A Compact Time Representation with Logarithmic Range	4
5. Protocol Integration of the Compact Time Representation	6
5.1. Interest Lifetime	7
5.2. Recommended Cache Time	8
6. IANA Considerations	8
7. Security Considerations	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Appendix A. Test Vectors	9
Acknowledgments	10
Authors' Addresses	10

1. Introduction

CCNx utilizes time values for a number of functions. Some of these are expressed as absolute time, others as delta time. When using CCNx in environments with constrained nodes and/or bandwidth constrained networks, it is valuable to have a compact representation of time values. For example [RFC9139] specifies a compression scheme useful over IEEE 802.15.4 networks. However, any compact time representation has to sacrifice either accuracy or dynamic range or both. For some time uses this is relatively straightforward to achieve, for other uses, it is not. This document discusses the various cases, and proposes a compact encoding that is easily accommodated for delta times.

2. Terminology

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].

This document uses the terminology of [RFC8569] and [RFC8609] for CCNx entities.

The following terms are used in the document and defined as follows:

byte: synonym for octet

time value: a time offset measured in seconds

time code: an 8-bit encoded time value

3. Usage of Time Values in CCNx

3.1. Relative Time in CCNx

CCNx, as currently specified in [RFC8569], utilizes delta time for only the lifetime of an Interest message (see sections 2.1, 2.2, 2.4.2, 10.3 of [RFC8569]). It is a hop-by-hop header value, and is currently encoded via the T_INTLIFE TLV as a 64-bit integer ([RFC8609] section 3.4.1). While formally an optional TLV, in all but some corner cases every Interest message is expected to carry the Interest Lifetime TLV, and hence having compact encoding is particularly valuable for keeping Interest messages short.

Since the current uses of delta time do not require both accuracy and dynamic range simultaneously, one can consider a logarithmic encoding such as that specified in [IEEE.754.2019] and outlined in Section 4. This document updates CCNx messages in TLV Format ([RFC8609]) to permit this alternative encoding for selected time values. See Section 6 for the specific actions needed to register this alternative compact representation of Interest Lifetime.

3.2. Absolute Time in CCNx

CCNx, as currently specified in [RFC8569], utilizes absolute time for various important functions. Each of these absolute time usages poses a different challenge for a compact representation. These are discussed in the following subsections.

3.2.1. Signature Time and Expiry Time

Signature Time is the time the signature of a content object was generated (sections 8.2–8.4 [RFC8569]). Expiry Time indicates the expiry time of a content object (section 4 [RFC8569]). Both values are content message TLVs and represent absolute timestamps in milliseconds since the UTC epoch (i.e., an NTP timestamp). They are currently encoded via the T_SIGTIME and T_EXPIRY TLVs as 64-bit unsigned integers (see section 3.6.4.1.4.5 [RFC8609] and section

3.6.2.2.2 [RFC8609]).

Both time values could be in the past, or in the future, potentially by a large delta. They are also included in the security envelope of the message. Therefore, it seems there is no practical way to define an alternative compact encoding that preserves its semantics and security properties; hence we don't consider it further as a candidate.

3.2.2. Recommended Cache Time

Recommended Cache Time (RCT) for a content object (see section 4 [RFC8569]) is a hop-by-hop header stating the expiration time for a cached content object in milliseconds since the UTC epoch (i.e., an NTP timestamp). It is currently encoded via the T_CACHETIME TLV as a 64-bit unsigned integer (see section 3.4.2 [RFC8609]).

A recommended cache time could be far in the future, but cannot be in the past and is likely to be a reasonably short offset from the current time. Therefore, this document allows the recommended cache time to be interpreted as a relative time value rather than an absolute time, since the semantics associated with an absolute time value do not seem to be critical to the utility of this value. This document therefore updates the recommended cache time with the following rule set:

- * Use absolute time as per [RFC8609]
- * Use relative time, if the compact time representation is used (see Section 4 and Section 5)

4. A Compact Time Representation with Logarithmic Range

This document uses the compact time representation of ICNLoWPAN (see section 7 of [RFC9139]) that is inspired by [RFC5497] and [IEEE.754.2019]. Its logarithmic encoding supports a representation ranging from milliseconds to years. Figure 1 depicts the logarithmic nature of this time representation.

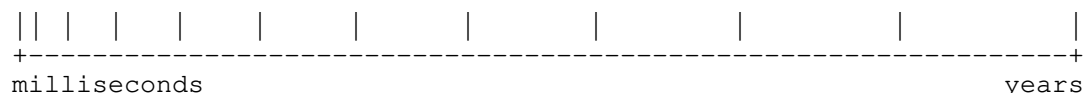


Figure 1: A logarithmic range representation allows for higher precision in the smaller time ranges and still supports large time deltas.

Time codes encode exponent and mantissa values in a single byte, but in contrast to the representation in [IEEE.754.2019], time codes only encode positive numbers and hence do not include an extra sign bit. Figure 2 shows the configuration of a time code: an exponent width of 5 bits, and a mantissa width of 3 bits.

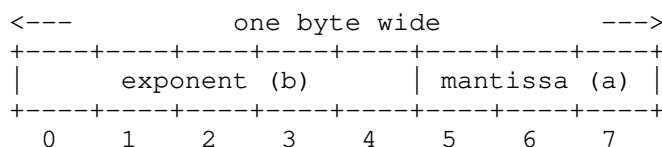


Figure 2: A time code with exponent and mantissa to encode a logarithmic range time representation.

The base unit for time values are seconds. A time value is calculated using the following formula (adopted from [RFC5497] and [RFC9139]), where (a) represents the mantissa, (b) the exponent, and (C) a constant factor set to $C := 1/32$.

Subnormal ($b == 0$): $(0 + a/8) * 2 * C$

Normalized ($b > 0$): $(1 + a/8) * 2^b * C$

The subnormal form provides a gradual underflow between zero and the smallest normalized number. Eight time values exist in the subnormal range $[0s, \sim 0.054688s]$ with a step size of $\sim 0.007812s$ between each time value. This configuration also encodes the following convenient numbers in seconds: $[1, 2, 4, 8, 16, 32, 64, \dots]$. Appendix A further includes test vectors to illustrate the logarithmic range.

An example algorithm to encode a time value into the corresponding exponent and mantissa is given as pseudo code in Figure 3. Not all time values can be represented by a time code. For these instances, the closest time code is chosen that is smaller than the value to encode.

```
input: float v    // time value
output: int a, b  // mantissa, exponent of time code

(a, b) encode (v):

    if (v == 0)
        return (0, 0)

    if (v < 2 * C)                                // subnormal
        a = floor (v * 4 / C)                     // round down
        return (a, 0)
    else                                           // normalized
        if (v > (1 + 7/8) * 2^31 * C)             // check bounds
            return (7, 31)                         // return maximum
        else
            b = floor (log2(v / C))                // round down
            a = floor ((v / (2^b * C) - 1) * 8)    // round down
            return (a, b)
```

Figure 3: Algorithm in pseudo code.

As an example: No specific time code for 0.063 exists, but this algorithm maps to the closest valid time code that is smaller, i.e., exponent 1 and mantissa 0 (the same as for time value 0.0625).

5. Protocol Integration of the Compact Time Representation

A straightforward way to accommodate the compact time approach is to use a 1-byte length field to indicate this alternative encoding while retaining the existing TLV registry entries. This approach has backward compatibility problems, but may still be considered for the following reasons:

- * Both CCNx RFCs are experimental and not Standards Track, hence expectations for forward and backward compatibility are not as stringent. "Flag day" upgrades of deployed CCNx networks, while inconvenient, are still feasible.
- * The major use case for these compressed encodings are smaller-scale IoT and/or sensor networks where the population of consumers, producers, and forwarders is reasonably small.
- * Since the current TLVs have hop-by-hop semantics, they are not covered by any signed hash and hence may be freely re-encoded by any forwarder. That means a forwarder supporting the new encoding can translate freely between the two encodings.

- * The alternative of assigning new TLV registry values does not substantially mitigate the interoperability problems anyway.

The following lists alternative approaches of integrating the compact time representation for time offsets in CCNx messages. A further analysis, discussion, and decision on the best suited approach will be added as the document progresses.

1. Relative time TLVs (e.g., T_INTLIFETIME) include nested TLVs to hint at the used encoding. This approach is the least intrusive integration, but adds a TLV overhead that negates the benefits of the compact time representation.
2. A new TLV type for T_INTLIFETIME with a compact time representation (T_INTLIFETIME_COMPACT) is defined. The packet header grammar from [RFC8609] is updated to allow for T_INTLIFETIME_COMPACT at the same level of the currently defined T_INTLIFETIME with an exclusive or.

5.1. Interest Lifetime

The Interest Lifetime definition in [RFC8609] allows for a variable-length lifetime representation, where a length of 1 encodes the linear range [0,255] in milliseconds. This document changes the definition to always encode 1-byte Interest lifetime values in the compact time value representation (Figure 4).

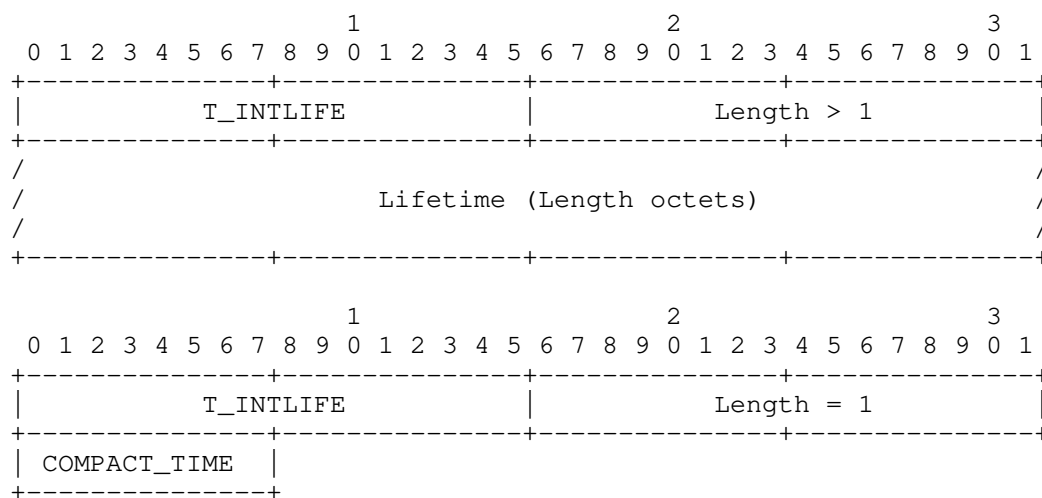


Figure 4: Changes to the definition of the Interest Lifetime TLV.

5.2. Recommended Cache Time

The Recommended Cache Time definition in [RFC8609] specifies an absolute time representation that is of a length fixed to 8 bytes. This document changes the definition to always encode 1-byte Recommended Cache Time values in the compact relative time value representation (Figure 5).

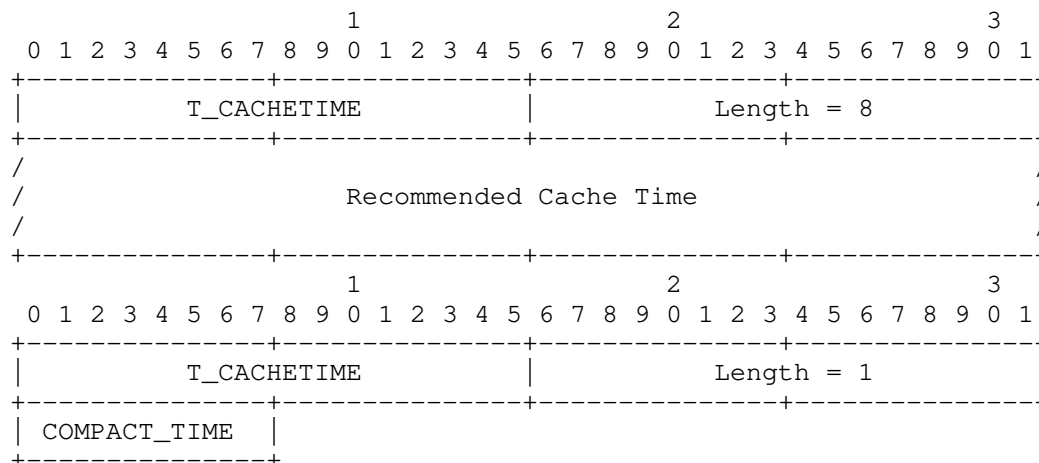


Figure 5: Changes to the definition of the Recommended Cache Time TLV.

The packet processing is adapted to calculate an absolute time from the relative time code based on the absolute reception time. On transmission, a new relative time code is calculated based on the current system time.

6. IANA Considerations

Based on the approach of integration, certain TLV registries from [RFC8609] need to be updated.

7. Security Considerations

This document makes no semantic changes to [RFC8569], nor to any of the security properties of the message encodings of [RFC8609], and hence has the same security considerations as those two existing documents.

8. References

8.1. Normative References

- [IEEE.754.2019] Institute of Electrical and Electronics Engineers, C/MSA - Microprocessor Standards Committee, "Standard for Floating-Point Arithmetic", June 2019, <<https://standards.ieee.org/content/ieee-standards/en/standard/754-2019.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, DOI 10.17487/RFC5497, March 2009, <<https://www.rfc-editor.org/info/rfc5497>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.

8.2. Informative References

- [RFC9139] Gündoan, C., Schmidt, T., Wählisch, M., Scherb, C., Marxer, C., and C. Tschudin, "Information-Centric Networking (ICN) Adaptation to Low-Power Wireless Personal Area Networks (LoWPANs)", RFC 9139, DOI 10.17487/RFC9139, November 2021, <<https://www.rfc-editor.org/info/rfc9139>>.

Appendix A. Test Vectors

The test vectors in Table 1 show sample time codes and their corresponding time values according to the algorithm outlined in Section 4.

Time Code	Time Value (seconds)
0x00	0.000000
0x01	0.007812
0x04	0.031250
0x08	0.062500
0x15	0.203125
0x28	1.000000
0x30	2.000000
0xF8	67108864.000000
0xFF	125829120.000000

Table 1: Test Vectors

Acknowledgments

We would like to thank the active members of the ICNRG research group for constructive thoughts. In particular, we thank Marc Mosko for his feedback on the encoding scheme, the provided pseudo code, and test vectors.

Authors' Addresses

Cenk Gundogan
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany

Phone: +4940428758067
Email: cenk.guendogan@haw-hamburg.de
URI: <http://inet.haw-hamburg.de/members/cenk-gundogan>

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany

Email: t.schmidt@haw-hamburg.de
URI: <http://inet.haw-hamburg.de/members/schmidt>

Dave Oran
Network Systems Research and Design
4 Shady Hill Square
Cambridge, MA 02138
United States of America

Email: daveoran@orandom.net

Matthias Waehlich
link-lab & FU Berlin
Hoenower Str. 35
D-10318 Berlin
Germany

Email: mw@link-lab.net
URI: <http://www.inf.fu-berlin.de/~waehl>

ICNRG
Internet-Draft
Intended status: Experimental
Expires: 11 May 2022

C. Tschudin
University of Basel
C.A. Wood
Cloudflare
M.E. Mosko
PARC, Inc.
D. Oran, Ed.
Network Systems Research & Design
7 November 2021

File-Like ICN Collections (FLIC)
draft-irtf-icnrg-flic-03

Abstract

This document describes a simple "index table" data structure and its associated ICN data objects for organizing a set of primitive ICN data objects into a large, File-Like ICN Collection (FLIC). At the core of this collection is a `_manifest_` which acts as the collection's root node. The manifest contains an index table with pointers, each pointer being a hash value pointing to either a final data block or another index table node.

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 <https://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 11 May 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	3
1.1. FLIC as an ICN experimental tool	5
1.2. Requirements Language	5
2. Design Overview	5
3. FLIC Structure	6
3.1. Terminology	6
3.2. Locators	8
3.3. Name Constructors	8
3.4. Manifest Metadata	10
3.5. Pointer Annotations	10
3.6. Manifest Grammar (ABNF)	11
3.7. Manifest Trees	13
3.7.1. Traversal	13
3.8. Manifest Encryption Modes	14
3.8.1. AEAD Mode	15
3.8.2. RSA-OAEP Key Transport Mode	17
3.9. Protocol Encodings	19
3.9.1. CCNx Encoding	19
3.9.1.1. CCNx Hash Naming	19
3.9.1.2. CCNx Single Prefix	20
3.9.1.3. CCNx Segmented Prefix	20
3.9.1.4. CCNx Hybrid Schema	21
3.9.2. NDN Encoding	22
3.9.2.1. NDN Hash Naming	22
3.9.2.2. NDN Single Prefix	22
3.9.2.3. NDN Segmented Prefix	23
3.9.2.4. NDN Hybrid Schema	24
3.10. Example Structures	25
3.10.1. Leaf-only data	25
3.10.2. Linear	25
4. Experimenting with FLIC	25
5. Usage Examples	26
5.1. Locating FLIC leaf and manifest nodes	26
5.2. Seeking	27
5.3. Block-level de-duplication	28
5.4. Growing ICN collections	28
5.5. Re-publishing a FLIC under a new name	29
6. IANA Considerations	30
6.1. FLIC Payload Type	30
6.2. FLIC Manifest Metadata and Annotation TLVs	30

7.	Security Considerations	31
7.1.	Integrity and Origin Authentication of FLIC Manifests . .	31
7.2.	Confidentiality of Manifest Data	32
7.3.	Privacy of names and linkability of access patterns . . .	33
8.	References	33
8.1.	Normative References	33
8.2.	Informative References	34
Appendix A.	Building Trees	35
Authors' Addresses	37

1. Introduction

ICN architectures such as Content-Centric Networking (CCNx) [RFC8569] and Named Data Networking [NDN] are well suited for static content distribution. Each piece of (possibly immutable) static content is assigned a name by its producer. Consumers fetch this content using said name. Optionally, consumers may specify the full name of content, which includes its name and a unique (with overwhelming probability) cryptographic digest of said content.

Note: The reader is assumed to be familiar with general ICN concepts from CCNx or NDN. For general ICN terms, this document uses the terminology defined in [RFC7927]. Where more specificity is needed, we utilize CCNx [RFC8569] terminology where a Content Object is the data structure that holds application payload. Terms defined specifically for FLIC are enumerated below in Section 3.1.

To enable requests with full names, consumers need a priori knowledge of content digests. Manifests, a form of catalog, are data structures commonly employed to store and transport this information. Typically, ICN manifests are signed content objects (data) which carry a collection of hash digests. Therefore, as content objects, manifests themselves may be fetched by full name. Thus, manifests may contain either hash digests of, or pointers to, either other manifests or content objects. A collection of manifests and content objects represents a large piece of application data, e.g., one that cannot otherwise fit in a single content object.

Structurally, this relationship between manifests and content objects is reminiscent of the UNIX inode concept with index tables and memory pointers. In this document, we specify a simple, yet extensible, manifest data structure called FLIC - _File-Like ICN Collection_. FLIC is suitable for ICN protocol suites such as CCNx and NDN. We describe the FLIC design, grammar, and various use cases, e.g., ordered fetch, seeking, de-duplication, extension, and variable-sized encoding. We also include FLIC encoding examples for CCNx and NDN.

The purpose of a manifest is to concisely name, and hence point to, the constituent pieces of a larger object. A FLIC manifest does this by using a `_root_` manifest to name and cryptographically sign the data structure and then use concise lists of hash-based names to indicate the constituent pieces. This maintains strong security from a single signature. A Manifest entry gives one enough information to create an `_Interest_` for that entry, so it must specify the name, the hash digest, and if needed, the locators.

FLIC is a distributed data structure illustrated by the following picture.

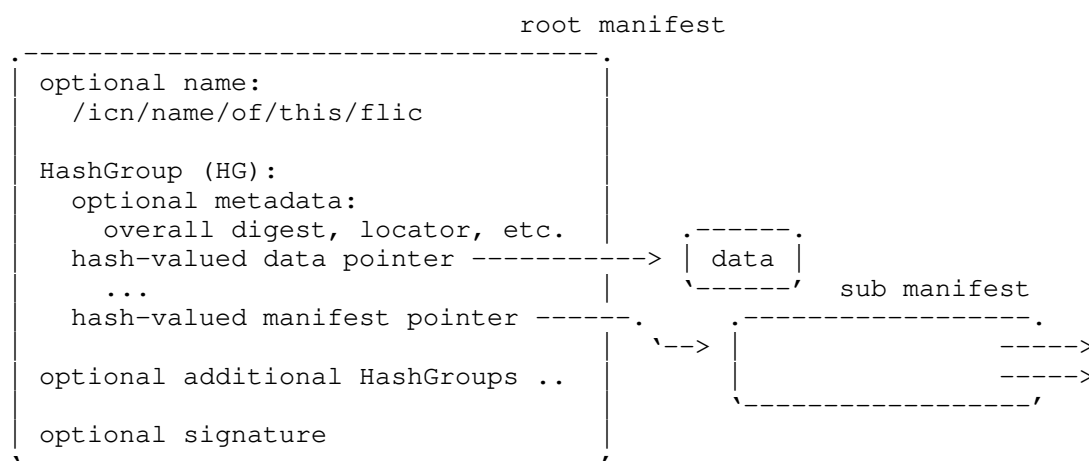


Figure 1: A FLIC manifest and its directed acyclic graph

A key design decision is how one names the root manifest, the application data, and subsidiary manifests. For this, FLIC uses the concept of a Name Constructor. The root manifest (in fact, any FLIC manifest) may include a Name Constructor that instructs a manifest reader how to properly create Interests for the associated application data and subsidiary manifests. The Name Constructors allow interest construction using a well-known, application-independent set of rules. Some name constructor forms are tailored towards specific ICN protocols, such as CCNx or NDN; some are more general and could work with many protocols. We describe the allowed Name Constructor methods in Section 3.3. There are also particulars of how to encode the name schema in a given ICN protocol, which we describe in Section 3.9.

FLIC has encodings for CCNx (Section 3.9.1) as per RFC 8609 [RFC8609] and for NDN (Section 3.9.2).

An example implementation in Python may be found at [FLICImplementation].

1.1. FLIC as an ICN experimental tool

FLIC enables experimentation with how to structure and retrieve large data objects and collections in ICN. By having a common data structure applications can rely on, with a common library of code that can be used to create and parse manifest data structures, applications using ICN protocols can both avoid unnecessary reinvention and also have enhanced interoperability. Since the design attempts to balance simplicity, universality, and extensibility, there are a number of important experimental goals to achieve that may wind up in conflict with one another. We provide a partial list of these experimental issues in Section 4. It is also important for users of FLIC to understand that some flexibility and extensions might be removed if use cases do not materialize to justify their inclusion in an eventual standard.

1.2. 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 [RFC2119].

2. Design Overview

The FLIC design adopts the proven UNIX inode concept of direct and indirect pointers, but without the specific structural forms of direct versus indirect. The pointers in FLIC use hash-based naming of Content Objects analogous to the function block numbers play in UNIX inodes. More generally, a FLIC structure is in most cases a tree, although any acyclic di-graph is a legal form.

In FLIC terms, a direct pointer links to application-level data, which is a Content Object with application data in the Payload. An indirect pointer links to a Content Object with a FLIC Manifest in the Payload.

Because FLIC uses hash-based pointers as names, FLIC graphs are inherently acyclic. Both CCNx and NDN support hash-based naming, though the details differ (see Section 3.9.1 and Section 3.9.2).

Note: A substantial advantage of using hash-based naming is that it permits block-level de-duplication of application data because two blocks with the same payload will have the same hash name.

The FLIC structure that it is expected most applications would use consists of a root manifest with a strong cryptographic signature and then cryptographically strong (e.g. SHA256 [SHS]) hash names as pointers to other manifests. The advantage of this structure is that the single signature in the root manifest covers the entire data structure no matter how many additional manifests are in the data structure. Another advantage of this structure is it removes the need to use chunk (CCNx) or segment (NDN) name components for the subordinate manifests.

FLIC supports manifest encryption separate from application payload encryption (See Section 3.8). It has a flexible encryption envelope to support various encryption algorithms and key discovery mechanisms. The byte layout allows for in-place encryption and decryption.

A limitation of this approach is that one cannot construct a hash-based name for a child until one knows the payload of that child. In practical terms, this means that one must have the complete application payload available at the time of manifest creation.

FLIC's design allows straightforward applications that just need to traverse a linear set of related objects to do so simply, but FLIC has two extensibility mechanisms that allow for more sophisticated uses: manifest metadata, and pointer annotations. These are described in Section 3.4 and Section 3.5 respectively.

FLIC goes to considerable lengths to allow creation and parsing by application-independent library code. Therefore, any options used by applications in the data structure or encryption capabilities MUST NOT require applications to have application-specific Manifest traversal algorithms. This ensures that such application agnostic libraries can always successfully parse and traverse any FLIC Manifest by ignoring the optional capabilities.

3. FLIC Structure

3.1. Terminology

Data Object: a CCNx nameless Content Object that usually only has Payload. It might also have an ExpiryTime to limit the lifetime of the data.

Direct Pointer: borrowed from inode terminology, it is a CCNx link using a content object hash restriction and a locator name to point to a Data Object.

Indirect Pointer: borrowed from inode terminology, it is a CCNx link

using a content object hash restriction and a locator name to point to a manifest content object.

Manifest: a CCNx ContentObject with PayloadType 'Manifest' and a Payload of the encoded manifest. A leaf manifest only has direct pointers. An internal manifest has a mixture of direct and indirect pointers.

Leaf Manifest: all pointers are direct pointers.

Internal Manifest: some or all pointers are indirect. The order and number of each is up to the manifest builder. By convention, all the direct manifests come first, then the indirect.

Manifest Waste: a metric used to measure the amount of waste in a manifest tree. Waste is the number of unused pointers. For example, a leaf manifest might be able to hold 40 direct pointers, but only 30 of them are used, so the waste of this node is 10. Manifest tree waste is the sum of waste over all manifests in a tree.

Root Manifest: A signed, named, manifest that points to nameless manifest nodes. This structure means that the internal tree structure of internal and leaf manifests have no names and thus may be located anywhere in a namespace, while the root manifest has a name to fetch it by.

Top Manifest: One useful manifest structure is to use a Root manifest that points to a single Internal manifest called the Top Manifest. The Top manifest the begins the structure used to organize manifests. It is also possible to elide the two and use only a root manifest that also serves in the role of the top manifest.

Name Constructor: The specification of how to construct an Interest for a Manifest entry.

Locator: A routing hint in an Interest used by forwarding to get the Interest to where it can be matched based on its Name Constructor-derived name.

3.2. Locators

Locators are routing hints used by forwarders to get an Interest to a node in the network that can resolve the Interest's name. In some naming conventions, the name might only be a hash-based name so the Locator is the only available routing information. Locators exist in both CCNx and NDN, though the specific protocol mechanisms differ. A FLIC manifest represents locators in the same way for both ICN protocols, though they are encoded differently in the underlying protocol. See Section 3.9 for encoding differences.

A manifest Node may define one or more Locator prefixes that can be used in the construction of Interests from the pointers in the manifest. The Locators are inherited when walking a manifest tree, so they do not need to be defined everywhere. It is RECOMMENDED that only the Root manifest contain Locators so that a single operation can update the locators. One use case when storing application payloads at different replicas is to replace the Root manifest with a new one that contains locators for the current replicas.

3.3. Name Constructors

A Manifest may define zero or more name constructors in NameConstructorDefinitions (NCD) located in the Manifest Node. An NCD associates a Name Constructor Id (NCID) to a Name Constructor. The NCID is used in other parts of the Manifest to refer to that specific definition.

NCID 0 is the default name constructor. If it is not defined in an NCD, it is assumed to be a HashNamingConstructor. A Manifest may re-define the default as needed.

A Manifest MUST use locally unique NCIDs in the NCD.

NCDs and their associated NCIDs are inherited as one traverses a manifest. That is, a manifest consumer must remember the NCDs as it traverses manifests. If it encounters a HashGroup that uses an unknown NCID, the RECOMMENDED action is to report a malformed manifest to the user.

A Manifest may update an NCID. If a child manifest re-defines an NCID, the manifest consumer MUST use the new definition from that point forward under that Manifest branch.

It is RECOMMENDED that only the root or similar top-level manifest define NCDs and they not be re-defined in subsequent manifests.

We expect that an application constructing a Manifest will take one of three approaches to name constructors. The advantage of using, or re-defining, the default name constructor is that any hash groups that use it do not need to specify an NCID and thus might save some space.

- * A manifest might define (or use) a default name constructor and mix subsequent Manifest and Data objects under that same namespace. The manifest only needs to use one Hash Group and can freely mix Manifest and Data pointers.
- * A manifest might define (or use) a default name constructor for subsequent Manifests and define a second NCD for the application data. This places all subsequent manifests under the default constructor and places all application data under the second NCD. The Manifest must use at least two Hash Groups.

There are a few options on how to organize the Hash Groups:

- (1) Manifest Hash Group followed by Data Hash group,
 - (2) Data Hash Group followed by Manifest Hash Group,
 - (3) Intermix multiple manifest and data hash groups for interleaved reading, or
 - (4) use a data-on-leaf only approach: the interior manifests would use the manifest hash group and the leaves would use the data hash group. Other organizations are possible.
- * Define multiple NCDs for subsequent manifests and data, or not use the default NCD, or use some other organization.

In this specification, we define the following four types of Name Constructors. Additional name constructor types may be specified in a subsequent revision of the specification. Here, we informally define the name constructors. Section 3.6 specifies the encoding of each name constructor.

Type 0 (Interest-Derived Naming): Use whatever name was used in the Interest to retrieve this Manifest, less a hash component, and append the desired hash value.

Type 1 (Data-Derived Naming): Use the Manifest Name, less a hash component, as the Interest name, and append the desired hash value.

Type 2 (Prefix List): The NCD specifies a list of 1 or more name prefixes. The consumer may use any (or all) of those prefixes with the desired hash appended.

Type 3 (Segmented Naming): As in Type 2, but the consumer MUST maintain a 0-based counter for each NCID associated with the in-order index of each hash and use that counter as a Segment number in the name.

3.4. Manifest Metadata

The FLIC Manifest may be extended by defining TLVs that apply to the Manifest as a whole, or alternatively, individually to every data object pointed to by the Manifest. This basic specification does not specify any, but metadata TLVs may be defined through additional RFCs or via Vendor TLVs. FLIC uses a Vendor TLV structure identical to [RFC8609] for vendor-specific annotations that require no standardization process.

For example, some applications may find it useful to allow specialized consumers such as `_repositories_` (for example [repository]) or enhanced forwarder caches to pre-place, or adaptively pre-fetch data in order to improve robustness and/or retrieval latency. Metadata can supply hints to such entities about what subset of the compound object to fetch and in what order.

Note: FLICs ability to use separate namespaces for the Manifest and the underlying Data allows different encryption keys to be used, hence giving a network element like a cache or repository access to the Manifest data does not as a side effect reveal the contents of the application data itself.

3.5. Pointer Annotations

FLIC allows each manifest pointer to be annotated with extra data. Annotations allow applications to exploit metadata about each Data Object pointed to without having to first fetch the corresponding Content Object. This specification defines one such annotation. The `_SizeAnnotation_` specifies the number of application layer octets covered by the pointer.

An annotation may, for example, give hints about a desirable traversal order for fetching the data, or an importance/precedence indication to aid applications that do not require every content object pointed to in the manifest to be fetched. This can be very useful for real-time or streaming media applications that can perform error concealment when rendering the media.

Additional annotations may be defined through additional RFCs or via Vendor TLVs. FLIC uses a Vendor TLV structure identical to [RFC8609] for vendor-specific annotations that require no standardization process.

3.6. Manifest Grammar (ABNF)

The manifest grammar is mostly, but not entirely independent of the ICN protocol used to encode and transport it. The TLV encoding therefore follows the corresponding ICN protocol, so for CCNx FLIC uses 2 octet length, 2 octet type and for NDN uses the 1/3/5 octet types and lengths (see [NDNTLV] for details). There are also some differences in how one structures and resolves links. [RFC8569] defines HashValue and Link for CCNx encodings. The NDN ImplicitSha256DigestComponent defines HashValue and NDN Delegation (from Link Object) defines Link for NDN. Section 3.9 below specifies these differences.

The basic structure of a FLIC manifest comprises a security context, a node, and an authentication tag. The security context and authentication tag are not needed if the node is unencrypted. A node is made up of a set of metadata, the NodeData, that applies to the entire node, and one or more HashGroups that contain pointers.

The NodeData element defines the namespaces used by the manifest. There may be multiple namespaces, depending on how one names subsequent manifests or data objects. Each HashGroup may reference a single namespace to control how one forms Interests from the HashGroup. If one is using separate namespaces for manifests and application data, one needs at least two hash groups. For a manifest structure of "MMMDDD," (where M means manifest (indirect pointer) and D means data (direct pointer)) for example, one would have a first HashGroup for the child manifests with its namespace and a second HashGroup for the data pointers with the other namespace. If one used a structure like "MMMDDMMM," then one would need three hash groups.

TYPE = 2OCTET / {1,3,5}OCTET ; As per CCNx or NDN TLV
 LENGTH = 2OCTET / {1,3,5}OCTET ; As per CCNx or NDN TLV

Manifest = TYPE LENGTH [SecurityCtx] (EncryptedNode / Node) [AuthTag]

SecurityCtx = TYPE LENGTH AlgorithmCtx
 AlgorithmCtx = AEADCtx / RsaKemCtx
 AuthTag = TYPE LENGTH *OCTET ; e.g. AEAD authentication tag
 EncryptedNode = TYPE LENGTH *OCTET ; Encrypted Node

Node = TYPE LENGTH [NodeData] 1*HashGroup
 NodeData = TYPE LENGTH [SubtreeSize] [SubtreeDigest] [Locators] 0*Vendor 0*NcD

ef

SubtreeSize = TYPE LENGTH INTEGER
 SubtreeDigest = TYPE LENGTH HashValue

NcDef = TYPE LENGTH NcId NcSchema

```

NcId = TYPE LENGTH INTEGER
NcSchema = InterestDerivedSchema / DataDerivedSchema / PrefixSchema / Segmente
dSchema
InterestDerivedSchema = TYPE LENGTH [ProtocolFlags]
DataDerivedSchema = TYPE LENGTH [ProtocolFlags]
PrefixSchema = TYPE LENGTH Locators [ProtocolFlags]
SegmentedSchema = TYPE LENGTH Locators [ProtocolFlags]

Locators = TYPE LENGTH 1*Link
HashValue = TYPE LENGTH *OCTET ; As per ICN Protocol
Link = TYPE LENGTH *OCTET ; As per ICN protocol

ProtocolFlags = TYPE LENGTH *OCTET; ICN-specific flags, e.g. must be fresh

HashGroup = TYPE LENGTH [GroupData] (Ptrs / AnnotatedPtrs)
Ptrs = TYPE LENGTH *HashValue
AnnotatedPtrs = TYPE LENGTH *PointerBlock
PointerBlock = TYPE LENGTH *Annotation Ptr
Ptr = TYPE LENGTH HashValue

Annotation = SizeAnnotation / Vendor
SizeAnnotation = TYPE LENGTH Integer
Vendor = TYPE LENGTH PEN *OCTET

GroupData = TYPE LENGTH [NcId] [LeafSize] [LeafDigest] [SubtreeSize] [SubtreeD
igest]
LeafSize = TYPE LENGTH INTEGER
LeafDigest = TYPE LENGTH HashValue

AEADCtx = TYPE LENGTH AEADData
AEADData = KeyNum AEADNonce Mode
KeyNum = TYPE LENGTH INTEGER
AEADNonce = TYPE LENGTH 1*OCTET
AEADMode = TYPE LENGTH (AEAD_AES_128_GCM / AEAD_AES_256_GCM / AEAD_AES_128_CCM
/ AEAD_AES_128_CCM)

RsaKemCtx = 2 LENGTH RsaKemData
RsaKemData = KeyId AEADNonce AEADMode WrappedKey LocatorPrefix
KeyId = TYPE LENGTH HashValue; ID of Key Encryption Key
WrappedKey = TYPE LENGTH 1*OCTET
LocatorPrefix = TYPE LENGTH Link

```

Figure 2: FLIC Grammar

SecurityCtx: information about how to decrypt an EncryptedNode. The structure will depend on the specific encryption algorithm.

AlgorithmId: The ID of the encryption method (e.g. preshared key, a broadcast encryption scheme, etc.)

AlgorithmData: The context for the encryption algorithm.

EncryptedNode: An opaque octet string with an optional authentication tag (i.e. for AEAD authentication tag)

Node: A plain-text manifest node. The structure allows for in-place encryption/decryption.

NodeData: the metadata about the Manifest node

SubtreeSize: The size of all application data at and below the Node or Group

SubtreeDigest: The cryptographic digest of all application data at and below the Node or Group

Locators: An array of routing hints to find the manifest components

HashGroup: A set of child pointers and associated metadata

Ptrs: A list of one or more Hash Values

GroupData: Metadata that applies to a HashGroup

LeafSize: Size of all application data immediately under the Group (i.e. via direct pointers)

LeafDigest: Digest of all application data immediately under the Group

Ptr: The ContentObjectHash of a child, which may be a data ContentObject (i.e. with Payload) or another Manifest Node.

3.7. Manifest Trees

3.7.1. Traversal

FLIC manifests use a pre-order traversal. This means they are read top to bottom, left to right. The algorithms in Figure 3 show the pre-order forward traversal code and the reverse-order traversal code, which we use below to construct such a tree. This document does not mandate how to build trees. Appendix A provides a detailed example of building inode-like trees.

If using Annotated Pointers, an annotation could influence the traversal order.


```

preorder(node)
  if (node = null)
    return
  visit(node)
  for (i = 0, i < node.child.length, i++)
    preorder(node.child[i])

reverse_preorder(node)
  if (node = null)
    return
  for (i = node.child.length - 1, i >= 0, i-- )
    reverse_preorder(node.child[i])
  visit(node)

```

Figure 3: Traversal Pseudocode

In terms of the FLIC grammar, one expands a node into its hash groups, visiting each hash group in order. In each hash group, one follows each pointer in order. Figure 4 shows how hash groups inside a manifest expand like virtual children in the tree. The in-order traversal is M0, HG1, M1, HG3, D0, D1, D2, HG2, D3, D4.

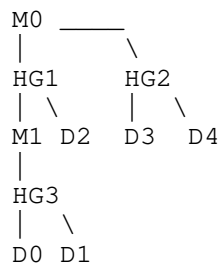


Figure 4: Node Expansion

Using the example manifest tree shown in Figure 6, the in-order traversal would be: Root, M0, M1, D0, D1, D2, M2, D3, D4, D5, M3, D6, D7, D8.

3.8. Manifest Encryption Modes

This document specifies two encryption modes. The first is a preshared key mode, where the parties are assumed to have the decryption keys already. It uses AES-GCM or AES-CCM. This is useful, for example, when using a key agreement protocol such as CCNxKE [I-D.wood-icnrg-ccnxkeyexchange]. The second is an RSA key encapsulation mode (RsaKem [RFC5990]), which may be used for group keying.

Additional modes may be defined in subsequent specifications. We expect that an RSA KemDem mode and Elliptic Curve mode should be specified.

All encryption modes use standard encryption algorithms and specifications. Where appropriate, we adopt the TLS 1.2 standards for how to use the encryption algorithms. This section specifies how to encode algorithm parameters or ICN-specific data.

For group key based encryption, we use RsaKem. This specification only details the pertinent aspects of the encryption. It describes how a consumer locates the appropriate keys in the ICN namespace. It does not specify aspects of a key manager which may or may not be used as part of key distribution and management, nor does it specify the protocol between a key manager and a publisher. In its simplest form, the publisher could be the key manager, in which case there is no extra protocol needed between the publisher and key manager.

While the preshared key algorithm is limited in use, the AES encryption mode described applies to the group key mechanisms too. The group key mechanism facilitates the distribution of the shared key without an on-line key agreement protocol like (the expired draft) CCNxKE [I-D.wood-icnrg-ccnxkeyexchange].

3.8.1. AEAD Mode

This mechanism uses AES-GEM [AESGCM] or AES-CCM [RFC3310] for manifest encryption. A publisher creating a SecurityCtx SHOULD use the mechanisms in [RFC6655] for AES-CCM Nonce generation and [RFC5288] for AES-GCM Nonce generation.

As these references specify, it is essential that the publisher creating a Manifest never use a Nonce more than once for the same key. For keys exchanged via a session protocol, such as CCNx, the publisher MUST use unique nonces on each Manifest for that session. If the key is derived via a group key mechanism, the publisher MUST ensure that the same Nonce is not used more than once for the same Content Encryption Key.

The AEAD Mode uses [RFC5116] defined symbols AEAD_AES_128_CCM, AEAD_AES_128_GCM, AEAD_AES_256_CCM and AEAD_AES_256_GCM to specify the key length and algorithm.

The KeyNum identifies a key on the receiver. The key MUST be exactly of the length specific by the Mode. Many receivers may have the same key with the same KeyNum.

When a Consumer reads a manifest that specifies a KeyNum, the consumer SHOULD verify that the Manifest's publisher is an expected one for the KeyNum's usage. This trust mechanism employed to ascertain whether the publisher is expected is beyond the scope of this document, but we provide an outline of one such possible trust mechanism. When a consumer learns a shared key and KeyNum, it associates that KeyNum with the publisher ID used in a public key signature. When the consumer receives a signed manifest (e.g. the root manifest of a manifest tree), the consumer matches the KeyNum's publisher with the Manifest's publisher.

Each encrypted manifest node has a full security context (KeyNum, Nonce, Mode). The AEAD decryption is independent for each manifest so Manifest objects can be fetched and decrypted in any order. This design also ensures that if a manifest tree points to the same subtree repeatedly, such as for deduplication, the decryptions are all idempotent.

To encrypt a Manifest, the publisher:

1. Removes any SecurityCtx or AuthTag from the Manifest.
2. Creates a SecurityCtx and adds it to the Manifest.
3. Treats the Manifest TLV through the end of the Node TLV Length as unencrypted authenticated Header. That includes anything from the start of the Manifest up to but not including the start of the Node's body.
4. Treats the body of the Node to the end of the Manifest as encrypted data.
5. Appends the AEAD AuthTag to the end of the Manifest, increasing the Manifest's length
6. Changes the TLV type of the Node to EncryptedNode.

To decrypt a Manifest, the consumer:

1. Verifies that the KeyNum exists and the publisher is trusted for that KeyNum.
2. Saves the AuthTag and removes it from the Manifest, decreasing the Manifest length.
3. Changes the EncryptedNode type to Node.

4. Treats everything from the Manifest TLV through the end of the Node Length as unencrypted authenticated Header. That is, all bytes from the start of the Manifest up to but not including the start of the Node's body.
5. Treats the body of the Node to the end of the Manifest as encrypted data.
6. Verifies and decrypts the data using the key and saved AuthTag.
7. If the decryption fails, the consumer SHOULD notify the user and stop further processing of the manifest.

3.8.2. RSA-OAEP Key Transport Mode

The RSA-OAEP mode uses RSA-OAEP (see RFC8017 Sec 7.1 [RFC8017] and [RSAKEM]) to encrypt a symmetric key that is used to encrypt the Manifest. We call this RSA key the Key Encryption Key (KEK) and each group member has this private key. A separate key distribution system is responsible for distributing the KEK. For our purposes, it is reasonable to assume that the KEK private key is available at a Locator and that group members can decrypt this private key.

The symmetric key MUST be one that is compatible with the AEAD Mode, i.e. a 128-bit or 256-bit random number. Further, the symmetric key MUST fit in the OAEP envelope (which will be true for normal-sized keys).

Any group key protocol and system needed are outside the scope of this document. We assume there is a Key Manager (KM) and a Publisher (P) and a set of group members. Through some means, the Publisher therefore has at its disposal:

- * A Content Encryption Key (CEK), i.e. the symmetric key.
- * The RSA-OAEP wrapped CEK.
- * The KeyId of the KEK used to wrap the CEK.
- * The Locator of the KEK, which is shared under some group key protocol.

This Manifest specification requires that if a group member fetches the KEK key at Locator it can decrypt the WrappedKey and retrieve the CEK.

In one example, a publisher could request a key for a group and the Key Manager could securely communicate (CEK, Wapped_CEK, KeyId, Locator) back to the publisher. The Key Manager is responsible for publishing the Locator. In another example, the publisher could be a group member and have a group private key in which case the publisher can create their own key encryption key, publish it under the Locator and proceed. The publisher generates CEK, Wrapped_CEK, KeyId, and a Locator on its own.

To create the wrapped key using a Key Encryption Key:

1. Obtain the CEK in binary format (e.g. 32 bytes for 256 bits)
2. RSA encrypt the CEK using the KEK public key with OAEP padding, following RFC8017 Sec 7.1 [RFC8017]. The encryption is not signed because the root Manifest must have been signed by the publisher already.

To decrypt the wrapped key using a Key Encryption Key:

1. RSA decrypt the WrappedKey using the KEK private key with OAEP padding, following RFC8017 Sec 7.1 [RFC8017].
2. Verify the unwrapped key is a valid length for the AEADMode.

To encrypt a Manifest, the publisher:

1. Acquires the set of (CEK, Wrapped_CEK, KeyId, Locator).
2. Creates a SecurityCtx and adds it to the Manifest. The SecurityCtx includes an AEADNonce and AEADMode, as per AEAD mode.
3. Encrypts the Manifest as per AEAD Mode using the RSA-OAEP SecurityCtx and CEK.

To decrypt a Manifest, the consumer:

1. Acquires the KEK from the Key Locator. If the consumer already has a cached copy of the KeyId in memory, it may use that cached key.
2. SHOULD verify that it trusts the Manifest publisher to use the provided key Locator.
3. Decrypts the WrappedKey to get the CEK. If the consumer has already decrypted the same exact WrappedKey TLV block, it may use that cached CEK.

4. Using the CEK, AEADNonce, and AEADMode, decrypt the Manifest as per AEAD Mode, ignoring the KeyNum steps.

3.9. Protocol Encodings

3.9.1. CCNx Encoding

In CCNx, application data content objects use a PayloadType of T_PAYLOADTYPE_DATA. In order to clearly distinguish FLIC Manifests from application data, a different payload type is required. Therefore this specification defines a new payload type of T_PAYLOADTYPE_FLIC.

```
ManifestContentObject = TYPE LENGTH [Name] [ExpiryTime] PayloadType Payload
Name = TYPE LENGTH *OCTET ; As per RFC8569
ExpiryTime = TYPE LENGTH *OCTET ; As per RFC8569
PayloadType = TYPE LENGTH T_PAYLOADTYPE_FLIC ; Value TBD
Payload : TYPE LENGTH *OCTET ; the serialized Manifest object
```

Figure 5: CCNx Embedding Grammar

3.9.1.1. CCNx Hash Naming

The Hash Naming namespace uses CCNx nameless content objects.

It proceeds as follows:

- * The Root Manifest content object bound to a name assigned by the publisher and signed by the publisher. It also may have a set of Locators used to fetch the remainder of the manifest. The root manifest has a single HashPointer that points to the Top Manifest. It may also have cache control directives, such as ExpiryTime.
- * The Root Manifest has an NsDef that specifies HashSchema. Its GroupData uses that NsId. All internal and leaf manifests use the same GroupData NsId. A Manifest Tree MAY omit the NsDef and NsId elements and rely on the default being HashSchema.
- * The Top Manifest is a nameless CCNx content object. It may have cache control directives.
- * Internal and Leaf manifests are nameless CCNx content objects, possibly with cache control directives.
- * The Data content objects are nameless CCNx content objects, possibly with cache control directives.

- * To form an Interest for a direct or indirect pointer, use a Name from one of the Locators and put the pointer HashValue into the ContentObjectHashRestriction.

3.9.1.2. CCNx Single Prefix

The Single Prefix schema uses the same name in all Content Objects and distinguishes them via their ContentObjectHash. Note that in CCNx, using a SinglePrefix name means that Locators are not used.

It proceeds as follows:

- * The Root Manifest content object has a name used to fetch the manifest. It is signed by the publisher. It has a set of Locators used to fetch the remainder of the manifest. It has a single HashPointer that points to the Top Manifest. It may also have cache control directives, such as ExpiryTime.
- * The Root Manifest has an NsDef that specifies SinglePrefix and the SinglePrefixSchema element specifies the SinglePrefixName.
- * The Top Manifest has the name SinglePrefixName. It may have cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * An Internal or Leaf manifest has the name SinglePrefixName, possibly with cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * The Data content objects have the name SinglePrefixName, possibly with cache control directives.
- * To form an Interest for a direct or indirect pointer, use SinglePrefixName as the Name and put the pointer HashValue into the ContentObjectHashRestriction.

3.9.1.3. CCNx Segmented Prefix

The Segmented Prefix schema uses a different name in all Content Objects and distinguishes them via their ContentObjectHash. Note that in CCNx, using a SegmentedPrefixSchema means that Locators are not used.

| *Optional*: Use AnnotatedPointers to indicate the segment
| number of each hash pointer to avoid needing to infer the
| segment numbers.

It proceeds as follows:

- * The Root Manifest content object has a name used to fetch the manifest. It is signed by the publisher. It has a set of Locators used to fetch the remainder of the manifest. It has a single HashPointer that points to the Top Manifest. It may also have cache control directives, such as ExpiryTime.
- * The Root Manifest has an NsDef that specifies SegmentedPrefix and the SegmentedPrefixSchema element specifies the SegmentedPrefixName.
- * The publisher tracks the chunk number of each content object within the NsId. Objects are be numbered in their traversal order. Within each manifest, the name can be constructed from the SegmentedPrefixName plus a Chunk name component.
- * The Top Manifest has the name SegmentedPrefixName plus chunk number. It may have cache control directives. It's GroupData elements must have an NsId that references the NsDef.
- * An Internal or Leaf manifest has the name SegmentedPrefixName plus chunk number, possibly with cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * The Data content objects have the name SegmentedPrefixName plus chunk number, possibly with cache control directives.
- * To form an Interest for a direct or indirect pointer, use SegmentedPrefixName plus chunk number as the Name and put the pointer HashValue into the ContentObjectHashRestriction. A consumer must track the chunk number in traversal order for each SegmentedPrefixSchema NsId.

3.9.1.4. CCNx Hybrid Schema

A manifest may use multiple schemas. For example, the application payload in data content objects might use SegmentedPrefix while the manifest content objects might use HashNaming.

The Root Manifest should specify an NsDef with a first NsId (say 1) as the HashNaming schema and a second NsDef with a second NsId (say 2) as the SegmentedPrefix schema along with the SegmentedPrefixName.

Each manifest (Top, Internal, Leaf) uses two or more HashGroups, where each HashGroup has only Direct (with the second NsId) or Indirect (with the first NsId). The number of hash groups will depend on how the publisher wishes to interleave direct and indirect pointers.

Manifests and data objects derive their names according to the application's naming schema.

3.9.2. NDN Encoding

In NDN, all Manifest Data objects use a ContentType of FLIC (1024), while all application data content objects use a PayloadType of Blob.

3.9.2.1. NDN Hash Naming

In NDN Hash Naming, a Data Object has a 0-length name. This means that an Interest will only have an ImplicitDigest name component in it. This method relies on using NDN Forwarding Hints.

It proceeds as follows:

- * The Root Manifest Data has a name used to fetch the manifest. It is signed by the publisher. It has a set of Locators used to fetch the remainder of the manifest. It has a single HashPointer that points to the Top Manifest. It may also have cache control directives.
- * The Root Manifest has an NsDef that specifies HashSchema. Its GroupData uses that NsId. All internal and leaf manifests use the same GroupData NsId. A Manifest Tree MAY omit the NsDef and NsId elements and rely on the default being HashSchema.
- * The Top Manifest has a 0-length Name. It may have cache control directives.
- * Internal and Leaf manifests has a 0-length Name, possibly with cache control directives.
- * The application Data use a 0-length name, possibly with cache control directives.
- * To form an Interest for a direct or indirect pointer, the name is only the Implicit Digest name component derived from a pointer's HashValue. The ForwardingHints come from the Locators. In NDN, one may use one or more locators within a single Interest.

3.9.2.2. NDN Single Prefix

In Single Prefix, the Data name is a common prefix used between all objects in that namespace, without a Segment or other counter. They are distinguished via the Implicit Digest name component. The FLIC Locators go in the ForwardingHints.

It proceeds as follows:

- * The Root Manifest Data object has a name used to fetch the manifest. It is signed by the publisher. It has a set of Locators used to fetch the remainder of the manifest. It has a single HashPointer that points to the Top Manifest. It may also have cache control directives.
- * The Root Manifest has an NsDef that specifies SinglePrefix and the SinglePrefixSchema element specifies the SinglePrefixName.
- * The Top Manifest has the name SinglePrefixName. It may have cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * An Internal or Leaf manifest has the name SinglePrefixName, possibly with cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * The Data content objects have the name SinglePrefixName, possibly with cache control directives.
- * To form an Interest for a direct or indirect pointer, use SinglePrefixName as the Name and append the pointer's HashValue into an ImplicitDigest name component. Set the ForwardingHints from the FLIC locators.

3.9.2.3. NDN Segmented Prefix

In Segmented Prefix, the Data name is a common prefix plus a segment number, so each manifest or application data object has a unique full name before the implicit digest. This means the consumer must maintain a counter for each SegmentedPrefix namespace.

| *Optional*: Use AnnotatedPointers to indicate the segment
| number of each hash pointer to avoid needing to infer the
| segment numbers.

It proceeds as follows:

- * The Root Manifest Data object has a name used to fetch the manifest. It is signed by the publisher. It has a set of Locators used to fetch the remainder of the manifest. It has a single HashPointer that points to the Top Manifest. It may also have cache control directives.

- * The Root Manifest has an NsDef that specifies SegmentedPrefix and the SegmentedPrefixSchema element specifies the SegmentedPrefixName.
- * The publisher tracks the segment number of each Data object within a SegmentedPrefix NsId. Data is numbered in traversal order. Within each manifest, the name is constructed from the SegmentedPrefixName plus a Segment name component.
- * The Top Manifest has the name SegmentedPrefixName plus segment number. It may have cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * An Internal or Leaf manifest has the name SegmentedPrefixName plus segment number, possibly with cache control directives. Its GroupData elements must have an NsId that references the NsDef.
- * The Data content objects have the name SegmentedPrefixName plus chunk number, possibly with cache control directives.
- * To form an Interest for a direct or indirect pointer, use SegmentedPrefixName plus segment number as the Name and put the pointer HashValue into the ImplicitDigest name component. A consumer must track the segment number in traversal order for each SegmentedPrefixSchema NsId.

3.9.2.4. NDN Hybrid Schema

A manifest may use multiple schemas. For example, the application payload in data content objects might use SegmentedPrefix while the manifest content objects might use HashNaming.

The Root Manifest should specify an NsDef with a first NsId (say 1) as the HashNaming schema and a second NsDef with a second NsId (say 2) as the SegmentedPrefix schema along with the SegmentedPrefixName.

Each manifest (Top, Internal, Leaf) uses two or more HashGroups, where each HashGroup has only Direct (with the second NsId) or Indirect (with the first NsId). The number of hash groups will depend on how the publisher wishes to interleave direct and indirect pointers.

Manifests and data objects derive their names according to the application's naming schema.

3.10. Example Structures

3.10.1. Leaf-only data

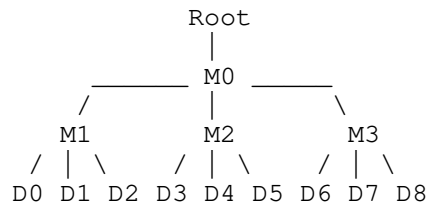


Figure 6: Leaf-only manifest tree

3.10.2. Linear

Of special interest are "skewed trees" where a pointer to a manifest may only appear as last pointer of (sub-) manifests. Such a tree becomes a sequential list of manifests with a maximum of datapointers per manifest packet. Beside the tree shape we also show this data structure in form of packet content where D stands for a data pointer and M is the hash of a manifest packet.

```

Root -> M0 ----> M1 ----> ...
|->DDDD |->DDDD

```

4. Experimenting with FLIC

FLIC is expected to enable a number of salient experiments in the use of ICN protocols by applications. These experiments will help not only to inform the desirable structure of ICN applications but reflect back to the features included in FLIC to evaluate their usefulness to those applications. While many interesting design aspects of FLIC remain to be discovered through experience, a number of important questions to be answered through experimentation include:

- * use for just files or other collections like directories
- * use for particular applications, like streaming media manifests
- * utility of pointer annotations to optimize retrieval
- * utility of the encryption options for use by repositories and forwarders
- * need for application metadata in manifests

5. Usage Examples

5.1. Locating FLIC leaf and manifest nodes

The names of manifest and data objects are often missing or not unique, unless using specific naming conventions. In this example, we show how using manifest locators is used to generate Interests. Take for example the figure below where the root manifest is named by hash h0. It has nameless children with hashes with hashes h1 ... hN.

```
Objects:
manifest(name=/a/b/c, ptr=h1, ptr=hN)  - has hash h0
nameless(data1)                        - has hash h1
...
nameless(dataN)                        - has hash hN
```

```
Query for the manifest:
interest(name=/a/b/c, implicitDigest=h0)
```

Figure 7: Data Organization

After obtaining the manifest, the client fetches the contents. In this first instance, the manifest does not provide any Locators data structure, so the client must continue using the name it used for the manifest.

```
interest(name=/a/b/c, implicitDigest=h1)
...
interest(name=/a/b/c, implicitDigest=hN)
```

Figure 8: Data Interests

Using the locator metadata entry, this behavior can be changed:

```
Objects:
manifest(name=/a/b/c,
hashgroup(loc=/x/y/z, ptr=h1)
hashgroup(ptr=h2)      - has hash h0
nameless(data1)        - has hash h1
nameless(data2)        - has hash h2
```

```
Queries:
interest(name=/a/b/c, implicitDigest=h0)
interest(name=/x/y/z, implicitDigest=h1)
interest(name=/a/b/c, implicitDigest=h2)
```

Figure 9: Using Locators

5.2. Seeking

Fast seeking (without having to sequentially fetch all content) works by skipping over entries for which we know their size. The following expression shows how to compute the byte offset of the data pointed at by pointer P_i , call it offset_i . In this formula, let $P_i.\text{size}$ represent the Size value of the i -th pointer.

$$\text{offset}_i = \sum_{k=1}^{i-1} P_k.\text{size}.$$

With this offset, seeking is done as follows:

Input: seek_pos P , a FLIC manifest with a hash group having N entries

Output: pointer index i and byte offset o , or out-of-range error

Algorithm:

```
offset = 0
for i in 1..N do
    if (P > offset + P_i.size)
        return (i, P - offset)
    offset += P_i.size
return out-of-range
```

Figure 10: Seeking Algorithm

Seeking in a BlockHashGroup is different since offsets can be quickly computed. This is because the size of each pointer P_i except the last is equal to the SizePerPtr value. For a BlockHashGroup with N pointers, OverallByteCount D , and SizePerPointer L , the size of P_N is equal to the following:

$$D - ((N - 1) * L)$$

In a BlockHashGroup with k pointers, the size of P_k is equal to:

$$D - L * (k - 1)$$

Using these, the seeking algorithm can be thus simplified to the following:

Input: seek_pos P, a FLIC manifest with a hash group having OverallByteCount S and SizePerPointer L.
 Output: pointer index i and byte offset o, or out-of-range error
 Algo:

```

    if (P > S)
        return out-of-range
    i = floor(P / L)
    if (i > N)
        return out-of-range # bad FLIC encoding
    o = P mod L
    return (i, o)

```

Figure 11: Seeking Algorithm

| *Note*: In both cases, if the pointer at position i is a
 | manifest pointer, this algorithm has to be called once more,
 | seeking to seek_pos o inside that manifest.

5.3. Block-level de-duplication

Consider a huge file, e.g. an ISO image of a DVD or program in binary be patched. In this case, all existing encoded ICN chunks can remain in the repository while only the chunks for the patch itself is added to a new manifest data structure, as is shown in the diagram below. For example, the venti archival file system of Plan9 [venti] uses this technique.

```

old_mfst - - > h1 --> oldData1 <-- h1 < - - new_mfst
          \ - > h2 --> oldData2 <-- h2 < - - /
            \          replace3 <-- h5 < - - /
              \- > h3 --> oldData3
                \ > h4 --> oldData4 <-- h4 < - /

```

Figure 12: De-duplication

5.4. Growing ICN collections

A log file, for example, grows over time. Instead of having to re-FLIC the grown file it suffices to construct a new manifest with a manifest pointer to the old root manifest plus the sequence of data hash pointers for the new data (or additional sub-manifests if necessary).

| *Note* that this tree will not be skewed (anymore).

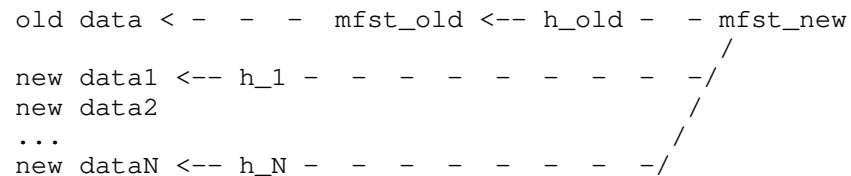


Figure 13: Growing A Collection

5.5. Re-publishing a FLIC under a new name

There are several use cases for republishing a collection under a new namespace, or having one collection exist under several namespaces:

- * It can happen that a publisher's namespace is part of a service provider's prefix. When switching provider, the publisher may want to republish the old data under a new name.
- * A publisher wishes to distribute its content to several repositories and would like a result to be delivered from the repository for consumers who have good connectivity to that repository. For example, the publisher /alpha wishes to place content at /beta and /gamma, but routing only to /alpha would not send a request to either /beta or /gamma. The operators of /beta and /gamma could create a named and signed version of the root manifest with appropriate keys (or delegate that to /alpha) so the results are always delivered by the corresponding repository without having to change the bulk of the manifest tree.

This can easily be achieved with a single nameless root manifest for the large FLIC plus arbitrarily many per-name manifests (which are signed by whomever wants to publish this data):

[illegible]

Figure 14: Relocating A Collection

Note that the hash computation (of h) only requires reading the nameless root manifest, not the entire FLIC.

This example points out the problem of HashGroups having their own locator metadata elements: A retriever would be urged to follow these hints which are "hardcoded" deep inside the FLIC but might have become outdated. We therefore recommend to name FLIC manifests only at the highest level (where these names have no locator function). Child nodes in a FLIC manifest should not be named as these names serve no purpose except retrieving a sub-tree's manifest by name, if would be required.

6. IANA Considerations

IANA is requested to perform the actions in the following sub-sections.

IANA should also note that FLIC uses the definitions of AEAD_AES_128_GCM, AEAD_AES_128_CCM, AEAD_AES_256_GCM, AEAD_AES_256_CCM from [RFC5116].

6.1. FLIC Payload Type

Register FLIC as a Payload Type in the `_CCNx Payload Types_` Registry referring to the description in Section 3.9.1 as follows:

Type	Name	Reference
TBA	T_PAYLOADTYPE_FLIC	Section 3.9.1 and Section 3.6.2.2.1 of [RFC8609]

Table 1: FLIC CCNx Payload Type

6.2. FLIC Manifest Metadata and Annotation TLVs

Create the following registry to be titled `_FLIC Manifest Metadata and Annotation TLVs_`. Manifest Metadata is described in Section 3.4; Pointer Annotations are described in Section 3.5. The registration procedure is **Specification Required**. The Type value is 2 octets. The range is 0x0000-0xFFFF. Allocate a value for the single `_SizeAnnotation_` TLV.

Type	Name	Reference
TBA	T_SIZE_ANNOTATION	Size (Section 3.5)

Table 2: FLIC Manifest Metadata and Annotation TLVs

7. Security Considerations

TODO Need a discussion on:

- * signing and hash chaining security. (*Note: Did I cover this adequately below?*)
 - * republishing under a new namespace. (*Note: need help here - is this to reinforce that you can re-publish application data by creating a new root Manifest and signing that, requiring only one signature to change?*)
 - * encryption mechanisms. (*Note: did I cover this adequately below?*)
 - * encryption key distribution mechanisms. (*Note: not sure what needs to be said here*)
 - * discussion of privacy, leaking of linkability information - *could really use some help here*.
- *Anything else????*.

7.1. Integrity and Origin Authentication of FLIC Manifests

A FLIC Manifest is used to describe how to form Interests to access large CCNx or NDN application data. The Manifest is itself either an individual content object, or a tree of content objects linked together via the corresponding content hashes. The NDN and CCNx protocol architectures directly provide both individual object integrity (using cryptographically strong hashes) and data origin authentication (using signatures). The protocol specifications, [NDN] and CCNx [RFC8609] respectively, provide the protocol machinery and keying to support strong integrity and authentication. Therefore, FLIC utilizes the existing protocol specifications for these functions, rather than providing its own. There are a few subtle differences in the handling of signatures and keys in NDN and CCNx worth recapitulating here:

- * NDN in general adds a signature to every individual data packet rather than aggregating signatures via some object-level scheme. When employing FLIC Manifests to multi-packet NDN objects, it is expected that the individual packet signatures would be elided and the signature on the Manifest used instead.
- * In contrast, CCNx is biased to have primitive objects or pieces thereof be "nameless" in the sense they are identified only by their hashes rather than each having a name directly bound to the content through an individual signature. Therefore, CCNx depends heavily on FLIC (or an alternative method) to provide the name and the signed binding of the name to the content described in the Manifest

A FLIC Manifest therefore gets integrity of its individual pieces through the existing secure hashing procedures of the underlying protocols. Origin authentication of the entire Manifest is achieved through hash chaining and applying a signature *only* to the root Manifest of a manifest tree. It is important to note that the Name of the Manifest, which is what the signature is bound to, need not bear any particular relationship to the names of the application objects pointed to in the Manifest via Name Constructors. This has a number of important benefits described in Section 3.3.

7.2. Confidentiality of Manifest Data

ICN protocol architectures like CCNx and NDN, while providing integrity and origin authentication as described above, leaves confidentiality issues entirely in the domain of the ICN application. Therefore, since FLIC is an application-level construct in both NDN and CCNx, it is incumbent on this specification for FLIC to provide the desired confidentiality properties using encryption. One could leave the specification of Manifest encryption entirely in the hands of the individual application utilizing FLIC, but this would be undesirable for a number of reasons:

- * The sensitivity of the information in a Manifest may be different from the sensitivity of the application data it describes. In some cases, it may not be necessary to encrypt manifests, or to encrypt them with a different keying scheme from that used for the application data
- * One of the major capabilities enabled by FLIC is to allow repositories or forwarding caches to operate on Manifests (see in particular Section 3.4). In order to allow such intermediaries to interpret manifests without revealing the underlying application data, separate encryption and keying is necessary

- * A strong design goal of FLIC is universality such that it can be used transparently by many different ICN applications. This argues that FLIC should have a set of common encryption and keying capabilities that can be delegated to library code and not have to be re-worked by each individual application (see Section 2, Paragraph 9)

Therefore, this specification directly specifies two encryption encapsulations and associated links to key management, as described in Section 3.8. As more experience is gained with various use cases, additional encryption capabilities may be needed and hence we expect the encryption aspects of this specification to evolve over time.

7.3. Privacy of names and linkability of access patterns

What to say here, if anything?

8. References

8.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3310] Niemi, A., Arkko, J., and V. Torvinen, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", RFC 3310, DOI 10.17487/RFC3310, September 2002, <<https://www.rfc-editor.org/info/rfc3310>>.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <<https://www.rfc-editor.org/info/rfc5116>>.
- [RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <<https://www.rfc-editor.org/info/rfc5288>>.
- [RFC5990] Randall, J., Kaliski, B., Brainard, J., and S. Turner, "Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS)", RFC 5990, DOI 10.17487/RFC5990, September 2010, <<https://www.rfc-editor.org/info/rfc5990>>.

- [RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <<https://www.rfc-editor.org/info/rfc6655>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.

8.2. Informative References

- [AESGCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", National Institute of Standards and Technology SP 800-38D, 2007, <<https://doi.org/10.6028/NIST.SP.800-38D>>.
- [FLICImplementation] Mosko, M., "FLIC Implementation in Python", various, <<https://github.com/mmosko/ccnpy>>.
- [I-D.wood-icnrg-ccnxkeyexchange] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", Work in Progress, Internet-Draft, draft-wood-icnrg-ccnxkeyexchange-02, 6 July 2017, <<https://datatracker.ietf.org/doc/html/draft-wood-icnrg-ccnxkeyexchange-02>>.
- [NDN] "Named Data Networking", various, <<https://named-data.net/project/execsummary/>>.
- [NDNTLV] "NDN Packet Format Specification.", 2016, <<http://named-data.net/doc/ndn-tlv/>>.
- [repository] "Repo Protocol Specification", Various, <https://redmine.named-data.net/projects/repo-ng/wiki/Repo_Protocol_Specification>.

- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.
- [RSAKEM] Barker, E., Chen, L., Roginsky, A., Vassilev, A., Davis, R., and S. Simon, "Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography", National Institute of Standards and Technology SP 800-56B Rev. 2, 2019, <<https://doi.org/10.6028/NIST.SP.800-56Br2>>.
- [SHS] Technology, N. I. O. S. A., "Secure Hash Standard, United States of American, National Institute of Science and Technology, Federal Information Processing Standard (FIPS) 180-4", National Institute of Standards and Technology SP 180-4, 2012, <https://csrc.nist.gov/publications/fips/fips180-4/fips180-4_final.pdf>.
- [venti] "Venti: a new approach to archival storage", Bell Labs Document Archive /sts/doc, 2002, <http://doc.cat-v.org/plan_9/4th_edition/papers/venti/>.

Appendix A. Building Trees

This appendix describes one method to build trees. It constructs a pre-order tree in a single pass of the application data, going from the tail to the beginning. This allows us to work up the right side of the tree in a single pass, then work down each left branch until we exhaust the data. Using the reverse-order traversal, we create the right-most-child manifest, then its parent, then the indirect pointers of that parent, then the parent's direct pointers, then the parent of the parent (repeating). This process uses recursion, as it is the clearest way to show the code. A more optimized approach could do it in a true single pass.

Because we're building from the bottom up, we use the term 'level' to be the distance from the right-most child up. Level 0 is the bottom-most level of the tree, such as where node 7 is:

```

      1
     2  3
    4 5  6 7
preorder: 1 2 4 5 3 6 7
reverse:  7 6 3 5 4 2 1

```

The Python-like pseudocode `build_tree(data, n, k, m)` algorithm creates a tree of n data objects. The `data[]` array is an array of Content Objects that hold application payload; the application data has already been packetized into n Content Object packets. An interior manifest node has k direct pointers and m indirect pointers.

```

build_tree(data[0..n-1], n, k, m)
    # data is an array of Content Objects (Data in NDN) with application payload.
    # n is the number of data items
    # k is the number of direct pointers per internal node
    # m is the number of indirect pointers per internal node

    segment = namedtuple('Segment', 'head tail')(0, n)
    level = 0

    # This bootstraps the process by creating the right most child manifest
    # A leaf manifest has no indirect pointers, so k+m are direct pointers
    root = leaf_manifest(data, segment, k + m)

    # Keep building subtrees until we're out of direct pointers
    while not segment.empty():
        level += 1
        root = bottom_up_preorder(data, segment, level, k, m, root)

    return root

bottom_up_preorder(data, segment, level, k, m, right_most_child=None)
    manifest = None
    if level == 0:
        assert right_most_child is None
        # build a leaf manifest with only direct pointers
        manifest = leaf_manifest(data, segment, k + m)
    else:
        # If the number of remaining direct pointers will fit in a leaf node,
        # make one of those.
        # Otherwise, we need to be an interior node
        if right_most_child is None and segment.length() <= k + m:
            manifest = leaf_manifest(data, segment, k+m)
        else:
            manifest = interior_manifest(data, segment, level, k, m, right_most_child)
    return manifest

leaf_manifest(data, segment, count)
    # At most count items, but never go before the head
    start = max(segment.head(), segment.tail() - count)
    manifest = Manifest(data[start:segment.tail()])
    segment.tail -= segment.tail() - start
    return manifest

```

```
interior_manifest(data, segment, level, k, m, right_most_child)
    children = []
    if right_most_child is not None:
        children.append(right_most_child)

    interior_indirect(data, segment, level, k, m, children)
    interior_direct(data, segment, level, k, m, children)

    manifest = Manifest(children)
    return manifest, tail

interior_indirect(data, segment, level, k, m, children)
    # Reserve space at the head of the segment for this node's direct pointers
before
    # descending to children. We want the top of the tree packed.
    reserve_count = min(k, segment.tail - segment.head)
    segment.head += reserve_count

    while len(children) < m and not segment.head == segment.tail:
        child = bottom_up_preorder(data, segment, level - 1, k, m)
        # prepend
        children.insert(0, child)

    # Pull back our reservation and put those pointers in our direct children
    segment.head -= reserve_count

interior_direct(data, segment, level, k, m, children)
    while len(children) < k+m and not segment.head == segment.tail:
        pointer = data[segment.tail() - 1]
        children.insert(0, pointer)
        segment.tail -= 1
```

Authors' Addresses

Christian Tschudin
University of Basel

Email: christian.tschudin@unibas.ch

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net

Marc Mosko
PARC, Inc.

Email: marc.mosko@parc.com

David Oran (editor)
Network Systems Research & Design

Email: daveoran@orandom.net

ICN Research Group
Internet-Draft
Intended status: Experimental
Expires: 31 March 2022

C. Gundogan
TC. Schmidt
HAW Hamburg
M. Waehlich
link-lab & FU Berlin
C. Scherb
C. Marxer
C. Tschudin
University of Basel
27 September 2021

ICN Adaptation to LoWPAN Networks (ICN LoWPAN)
draft-irtf-icnrg-icnlowpan-11

Abstract

This document defines a convergence layer for CCNx and NDN over IEEE 802.15.4 LoWPAN networks. A new frame format is specified to adapt CCNx and NDN packets to the small MTU size of IEEE 802.15.4. For that, syntactic and semantic changes to the TLV-based header formats are described. To support compatibility with other LoWPAN technologies that may coexist on a wireless medium, the dispatching scheme provided by 6LoWPAN is extended to include new dispatch types for CCNx and NDN. Additionally, the fragmentation component of the 6LoWPAN dispatching framework is applied to ICN chunks. In its second part, the document defines stateless and stateful compression schemes to improve efficiency on constrained links. Stateless compression reduces TLV expressions to static header fields for common use cases. Stateful compression schemes elide state local to the LoWPAN and replace names in data packets by short local identifiers.

This document is a product of the IRTF Information-Centric Networking Research Group (ICNRG).

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 <https://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 31 March 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	3
2. Terminology	5
3. Overview of ICN LoWPAN	6
3.1. Link-Layer Convergence	6
3.2. Stateless Header Compression	6
3.3. Stateful Header Compression	8
4. IEEE 802.15.4 Adaptation	8
4.1. LoWPAN Encapsulation	8
4.1.1. Dispatch Extensions	9
4.2. Adaptation Layer Fragmentation	10
5. Space-efficient Message Encoding for NDN	11
5.1. TLV Encoding	11
5.2. Name TLV Compression	13
5.3. Interest Messages	13
5.3.1. Uncompressed Interest Messages	13
5.3.2. Compressed Interest Messages	14
5.3.3. Dispatch Extension	16
5.4. Data Messages	17
5.4.1. Uncompressed Data Messages	17
5.4.2. Compressed Data Messages	17
5.4.3. Dispatch Extension	19
6. Space-efficient Message Encoding for CCNx	20
6.1. TLV Encoding	20
6.2. Name TLV Compression	20
6.3. Interest Messages	20
6.3.1. Uncompressed Interest Messages	20
6.3.2. Compressed Interest Messages	21
6.3.3. Dispatch Extension	26

6.4. Content Objects	27
6.4.1. Uncompressed Content Objects	27
6.4.2. Compressed Content Objects	27
6.4.3. Dispatch Extension	30
7. Compressed Time Encoding	31
8. Stateful Header Compression	32
8.1. LoWPAN-local State	32
8.2. En-route State	32
8.3. Integrating Stateful Header Compression	34
9. ICN LoWPAN Constants and Variables	35
10. Implementation Report and Guidance	35
10.1. Preferred Configuration	35
10.2. Further Experimental Deployments	36
11. Security Considerations	37
12. IANA Considerations	37
12.1. Reserving Space in the 6LoWPAN Dispatch Type Field Registry	38
13. References	38
13.1. Normative References	38
13.2. Informative References	39
Appendix A. Estimated Size Reduction	42
A.1. NDN	42
A.1.1. Interest	42
A.1.2. Data	43
A.2. CCNx	45
A.2.1. Interest	45
A.2.2. Content Object	46
Acknowledgments	47
Authors' Addresses	47

1. Introduction

The Internet of Things (IoT) has been identified as a promising deployment area for Information Centric Networks (ICN), as infrastructureless access to content, resilient forwarding, and in-network data replication demonstrated notable advantages over the traditional host-to-host approach on the Internet [NDN-EXP1], [NDN-EXP2]. Recent studies [NDN-MAC] have shown that an appropriate mapping to link layer technologies has a large impact on the practical performance of an ICN. This will be even more relevant in the context of IoT communication where nodes often exchange messages via low-power wireless links under lossy conditions. In this memo, we address the base adaptation of data chunks to such link layers for the ICN flavors NDN [NDN] and CCNx [RFC8569], [RFC8609].

The IEEE 802.15.4 [ieee802.15.4] link layer is used in low-power and lossy networks (see LLN in [RFC7228]), in which devices are typically battery-operated and constrained in resources. Characteristics of

LLNs include an unreliable environment, low bandwidth transmissions, and increased latencies. IEEE 802.15.4 admits a maximum physical layer packet size of 127 bytes. The maximum frame header size is 25 bytes, which leaves 102 bytes for the payload. IEEE 802.15.4 security features further reduce this payload length by up to 21 bytes, yielding a net of 81 bytes for CCNx or NDN packet headers, signatures and content.

6LoWPAN [RFC4944], [RFC6282] is a convergence layer that provides frame formats, header compression and adaptation layer fragmentation for IPv6 packets in IEEE 802.15.4 networks. The 6LoWPAN adaptation introduces a dispatching framework that prepends further information to 6LoWPAN packets, including a protocol identifier for payload and meta information about fragmentation.

Prevalent Type-Length-Value (TLV) based packet formats such as in CCNx and NDN are designed to be generic and extensible. This leads to header verbosity which is inappropriate in constrained environments of IEEE 802.15.4 links. This document presents ICN LoWPAN, a convergence layer for IEEE 802.15.4 motivated by 6LoWPAN. ICN LoWPAN compresses packet headers of CCNx as well as NDN and allows for an increased effective payload size per packet. Additionally, reusing the dispatching framework defined by 6LoWPAN enables compatibility between coexisting wireless deployments of competing network technologies. This also allows to reuse the adaptation layer fragmentation scheme specified by 6LoWPAN for ICN LoWPAN.

ICN LoWPAN defines a more space efficient representation of CCNx and NDN packet formats. This syntactic change is described for CCNx and NDN separately, as the header formats and TLV encodings differ notably. For further reductions, default header values suitable for constrained IoT networks are selected in order to elide corresponding TLVs. Experimental evaluations of the ICN LoWPAN header compression schemes in [ICNLOWPAN] illustrate a reduced message overhead, a shortened message airtime, and an overall decline in power consumption for typical Class 2 [RFC7228] devices compared to uncompressed ICN messages.

In a typical IoT scenario (see Figure 1), embedded devices are interconnected via a quasi-stationary infrastructure using a border router (BR) that connects the constrained LoWPAN network by some Gateway with the public Internet. In ICN based IoT networks, non-local Interest and Data messages transparently travel through the BR up and down between a Gateway and the embedded devices situated in the constrained LoWPAN.

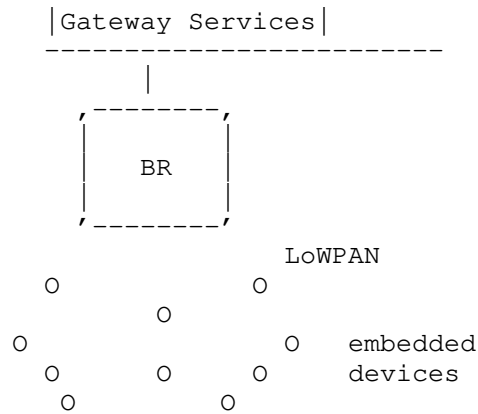


Figure 1: IoT Stub Network

The document has received fruitful reviews by members of the ICN community and the research group (see Acknowledgments) for a period of two years. It is the consensus of ICNRG that this document should be published in the IRTF Stream of the RFC series. This document does not constitute an IETF standard.

2. Terminology

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]. The use of the term, "silently ignore" is not defined in RFC 2119. However, the term is used in this document and can be similarly construed.

This document uses the terminology of [RFC7476], [RFC7927], and [RFC7945] for ICN entities.

The following terms are used in the document and defined as follows:

ICN LoWPAN:	Information-Centric Networking over Low-power Wireless Personal Area Network
LLN:	Low-Power and Lossy Network
CCNx:	Content-Centric Networking Architecture
NDN:	Named Data Networking Architecture
byte:	synonym for octet

nibble: synonym for 4 bits

time-value: a time offset measured in seconds

time-code: an 8-bit encoded time-value

3. Overview of ICN LoWPAN

3.1. Link-Layer Convergence

ICN LoWPAN provides a convergence layer that maps ICN packets onto constrained link-layer technologies. This includes features such as link-layer fragmentation, protocol separation on the link-layer level, and link-layer address mappings. The stack traversal is visualized in Figure 2.

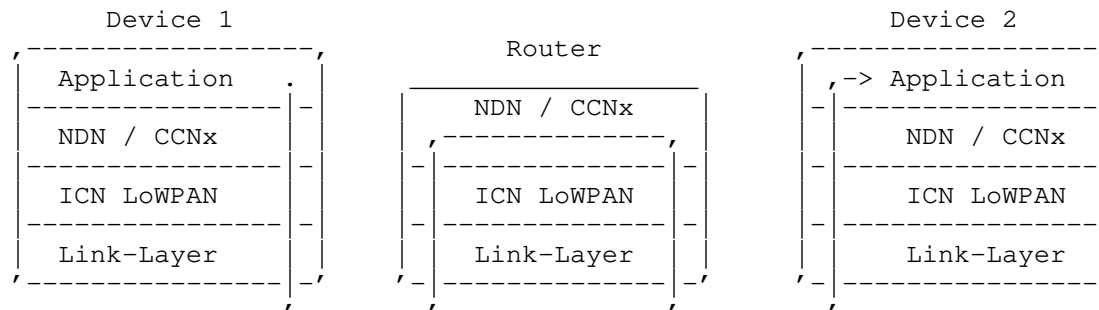


Figure 2: ICN LoWPAN convergence layer for IEEE 802.15.4

Section 4 of this document defines the convergence layer for IEEE 802.15.4.

3.2. Stateless Header Compression

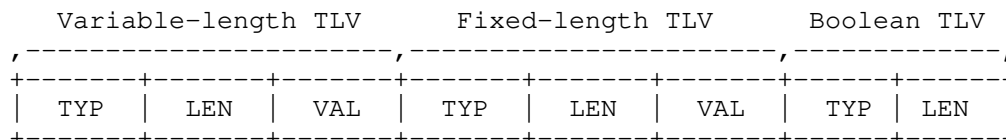
ICN LoWPAN also defines a stateless header compression scheme with the main purpose of reducing header overhead of ICN packets. This is of particular importance for link-layers with small MTUs. The stateless compression does not require pre-configuration of global state.

The CCNx and NDN header formats are composed of Type-Length-Value (TLV) fields to encode header data. The advantage of TLVs is its native support of variably structured data. The main disadvantage of TLVs is the verbosity that results from storing the type and length of the encoded data.

The stateless header compression scheme makes use of compact bit fields to indicate the presence of optional TLVs in the uncompressed packet. The order of set bits in the bit fields corresponds to the order of each TLV in the packet. Further compression is achieved by specifying default values and reducing the range of certain header fields.

Figure 3 demonstrates the stateless header compression idea. In this example, the first type of the first TLV is removed and the corresponding bit in the bit field is set. The second TLV represents a fixed-length TLV (e.g., the Nonce TLV in NDN), so that the type and the length fields are removed. The third TLV represents a boolean TLV (e.g., the MustBeFresh selector in NDN) for which the type, length and the value fields are elided.

Uncompressed:



Compressed:

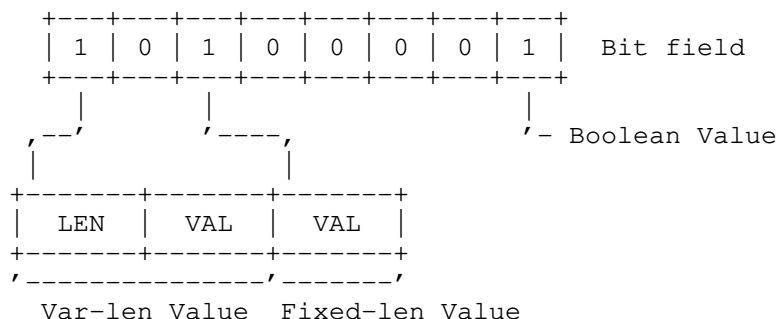


Figure 3: Compression using a compact bit field - bits encode the inclusion of TLVs.

Stateless TLV compression for NDN is defined in Section 5. Section 6 defines the stateless TLV compression for CCNx.

The extensibility of this compression is described in Section 4.1.1 and allows future documents to update the compression rules outlined in this manuscript.

3.3. Stateful Header Compression

ICN LoWPAN further employs two orthogonal stateful compression schemes for packet size reductions which are defined in Section 8. These mechanisms rely on shared contexts that are either distributed and maintained in the entire LoWPAN, or are generated on-demand hop-wise on a particular Interest-data path.

The shared context identification is defined in Section 8.1. The hop-wise name compression "en-route" is specified in Section 8.2.

4. IEEE 802.15.4 Adaptation

4.1. LoWPAN Encapsulation

The IEEE 802.15.4 frame header does not provide a protocol identifier for its payload. This causes problems of misinterpreting frames when several network layers coexist on the same link. To mitigate errors, 6LoWPAN defines dispatches as encapsulation headers for IEEE 802.15.4 frames (see Section 5 of [RFC4944]). Multiple LoWPAN encapsulation headers can precede the actual payload and each encapsulation header is identified by a dispatch type.

[RFC8025] further specifies dispatch pages to switch between different contexts. When a LoWPAN parser encounters a Page switch LoWPAN encapsulation header, then all following encapsulation headers are interpreted by using a dispatch table as specified by the Page switch header. Page 0 and page 1 are reserved for 6LoWPAN. This document uses page TBD1 (1111 TBD1 (0xFTBD1)) for ICN LoWPAN.

The base dispatch format (Figure 4) is used and extended by CCNx and NDN in Section 5 and Section 6.

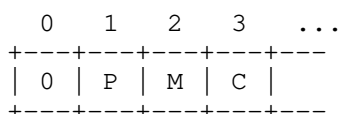


Figure 4: Base dispatch format for ICN LoWPAN

P: Protocol 0: The included protocol is NDN.

 1: The included protocol is CCNx.

M: Message Type 0: The payload contains an Interest message.

 1: The payload contains a Data message.

C: Compression 0: The message is uncompressed.

1: The message is compressed.

ICN LoWPAN frames with compressed CCNx and NDN messages (C=1) use the extended dispatch format in Figure 5.

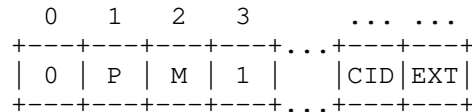


Figure 5: Extended dispatch format for compressed ICN LoWPAN

CID: Context Identifier 0: No context identifiers are present.

1: Context identifier(s) are present (see Section 8.1).

EXT: Extension 0: No extension bytes are present.

1: Extension byte(s) are present (see Section 4.1.1).

The encapsulation format for ICN LoWPAN is displayed in Figure 6.

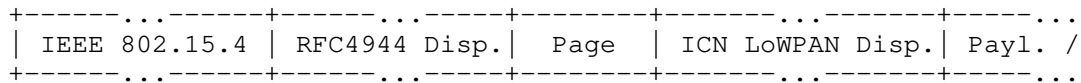


Figure 6: LoWPAN Encapsulation with ICN-LoWPAN

IEEE 802.15.4: The IEEE 802.15.4 header.

RFC4944 Disp.: Optional additional dispatches defined in Section 5.1 of [RFC4944]

Page: Page Switch. TBD1 for ICN LoWPAN.

ICN LoWPAN: Dispatches as defined in Section 5 and Section 6.

Payload: The actual (un-)compressed CCNx or NDN message.

4.1.1. Dispatch Extensions

Extension bytes allow for the extensibility of the initial compression rule set. The base format for an extension byte is depicted in Figure 7.

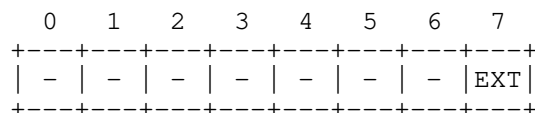


Figure 7: Base format for dispatch extensions.

EXT: Extension 0: No other extension byte follows.

1: A further extension byte follows.

Extension bytes are numbered according to their order. Future documents MUST follow the naming scheme EXT_0, EXT_1, ..., when updating or referring to a specific dispatch extension byte. Amendments that require an exchange of configurational parameters between devices SHOULD use manifests to encode structured data in a well-defined format, as, e.g., outlined in [I-D.irtf-icnrg-flic].

4.2. Adaptation Layer Fragmentation

Small payload sizes in the LoWPAN require fragmentation for various network layers. Therefore, Section 5.3 of [RFC4944] defines a protocol-independent fragmentation dispatch type, a fragmentation header for the first fragment, and a separate fragmentation header for subsequent fragments. ICN LoWPAN adopts this fragmentation handling of [RFC4944].

The Fragmentation LoWPAN header can encapsulate other dispatch headers. The order of dispatch types is defined in Section 5 of [RFC4944]. Figure 8 shows the fragmentation scheme. The reassembled ICN LoWPAN frame does not contain any fragmentation headers and is depicted in Figure 9.

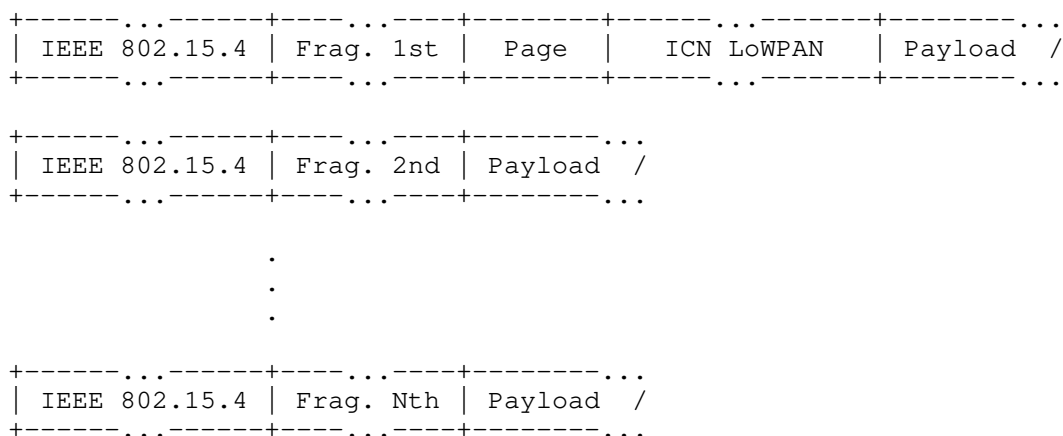


Figure 8: Fragmentation scheme

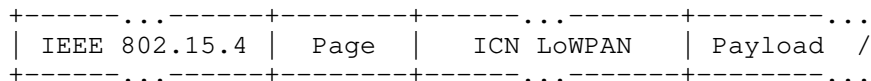


Figure 9: Reassembled ICN LoWPAN frame

The 6LoWPAN Fragment Forwarding (6FF) [RFC8930] is an alternative approach that enables forwarding of fragments without reassembling packets on every intermediate hop. By reusing the 6LoWPAN dispatching framework, 6FF integrates into ICN LoWPAN as seamless as the conventional hop-wise fragmentation. Experimental evaluations [SFR-ICNLOWPAN], however, suggest that a more refined integration can increase the cache utilization of forwarders on a request path.

5. Space-efficient Message Encoding for NDN

5.1. TLV Encoding

The NDN packet format consists of TLV fields using the TLV encoding that is described in [NDN-PACKET-SPEC]. Type and length fields are of variable size, where numbers greater than 252 are encoded using multiple bytes.

If the type or length number is less than 253, then that number is encoded into the actual type or length field. If the number is greater or equals 253 and fits into 2 bytes, then the type or length field is set to 253 and the number is encoded in the next following 2 bytes in network byte order, i.e., from the most significant byte (MSB) to the least significant byte (LSB). If the number is greater than 2 bytes and fits into 4 bytes, then the type or length field is set to 254 and the number is encoded in the subsequent 4 bytes in network byte order. For larger numbers, the type or length field is set to 255 and the number is encoded in the subsequent 8 bytes in network byte order.

In this specification, compressed NDN TLVs encode type and length fields using self-delimiting numeric values (SDNVs) [RFC6256] commonly known from DTN protocols. Instead of using the first byte as a marker for the number of following bytes, SDNVs use a single bit to indicate subsequent bytes.

Value	NDN TLV encoding	SDNV encoding
0	0x00	0x00
127	0x7F	0x7F
128	0x80	0x81 0x00
253	0xFD 0x00 0xFD	0x81 0x7D
$2^{14} - 1$	0xFD 0x3F 0xFF	0xFF 0x7F
2^{14}	0xFD 0x40 0x00	0x81 0x80 0x00
2^{16}	0xFE 0x00 0x01 0x00 0x00	0x84 0x80 0x00
$2^{21} - 1$	0xFE 0x00 0x1F 0xFF 0xFF	0xFF 0xFF 0x7F
2^{21}	0xFE 0x00 0x20 0x00 0x00	0x81 0x80 0x80 0x00
$2^{28} - 1$	0xFE 0x0F 0xFF 0xFF 0xFF	0xFF 0xFF 0xFF 0x7F
2^{28}	0xFE 0x1F 0x00 0x00 0x00	0x81 0x80 0x80 0x80 0x00
2^{32}	0xFF 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00	0x90 0x80 0x80 0x80 0x00
$2^{35} - 1$	0xFF 0x00 0x00 0x00 0x07 0xFF 0xFF 0xFF 0xFF	0xFF 0xFF 0xFF 0xFF 0x7F
2^{35}	0xFF 0x00 0x00 0x00 0x08 0x00 0x00 0x00 0x00	0x81 0x80 0x80 0x80 0x80 0x00

Table 1: NDN TLV encoding compared to SDNVs.

Table 1 compares the required bytes for encoding a few selected values using the NDN TLV encoding and SDNVs. For values up to 127, both methods require a single byte. Values in the range [128;252] encode as one byte for the NDN TLV scheme, while SDNVs require two bytes. Starting at value 253, SDNVs require a less or equal amount of bytes compared to the NDN TLV encoding.

5.2. Name TLV Compression

This Name TLV compression encodes length fields of two consecutive NameComponent TLVs into one byte, using a nibble for each. The most significant nibble indicates the length of an immediately following NameComponent TLV. The least significant nibble denotes the length of a subsequent NameComponent TLV. A length of 0 marks the end of the compressed Name TLV. The last length field of an encoded NameComponent is either 0x00 for a name with an even number of components, and 0xYF (Y > 0) if an odd number of components are present. This process limits the length of a NameComponent TLV to 15 bytes, but allows for an unlimited number of components. An example for this encoding is presented in Figure 10.

Name: /HAW/Room/481/Humid/99

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	0	0	1	1		0	1	0	0																						
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	0	0	1	1		0	1	0	1																						
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Figure 10: Name TLV compression for /HAW/Room/481/Humid/99

5.3. Interest Messages

5.3.1. Uncompressed Interest Messages

An uncompressed Interest message uses the base dispatch format (see Figure 4) and sets the C flag to 0 and the P as well as the M flag to 0 (Figure 11). The Interest message is handed to the NDN network stack without modifications.

0	1	2	3	4	5	6	7
+	+	+	+	+	+	+	+
	0		0		0		0
+	+	+	+	+	+	+	+

Figure 11: Dispatch format for uncompressed NDN Interest messages

5.3.2. Compressed Interest Messages

The compressed Interest message uses the extended dispatch format (Figure 5) and sets the C flag to 1, the P flag to 0 and the M flag to 0. If an Interest message contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

This specification assumes that a HopLimit TLV is part of the original Interest message. If such HopLimit TLV is not present, it will be inserted with a default value of DEFAULT_NDN_HOPLIMIT prior to the compression.

In the default use case, the Interest message is compressed with the following minimal rule set:

1. The Type field of the outermost MessageType TLV is removed.
2. The Name TLV is compressed according to Section 5.2. For this, all NameComponents are expected to be of type GenericNameComponent with a length greater than 0. An ImplicitSha256DigestComponent or ParametersSha256DigestComponent MAY appear at the end of the name. In any other case, the message MUST be sent uncompressed.
3. The Nonce TLV and InterestLifetime TLV are moved to the end of the compressed Interest as illustrated in Figure 12. The InterestLifetime is encoded as described in Section 7. On decompression, this encoding may yield an InterestLifetime that is smaller than the original value.
4. The Type and Length fields of Nonce TLV, HopLimit TLV and InterestLifetime TLV are elided. The Nonce value has a length of 4 bytes and the HopLimit value has a length of 1 byte. The compressed InterestLifetime (Section 7) has a length of 1 byte. The presence of a Nonce and InterestLifetime TLV is deduced from the remaining length to parse. A remaining length of 1 indicates the presence of an InterestLifetime, a length of 4 indicates the presence of a nonce, and a length of 5 indicates the presence of both TLVs.

The compressed NDN LoWPAN Interest message is visualized in Figure 12.

T = Type, L = Length, V = Value
 Lc = Compressed Length, Vc = Compressed Value
 : = optional field, | = mandatory field

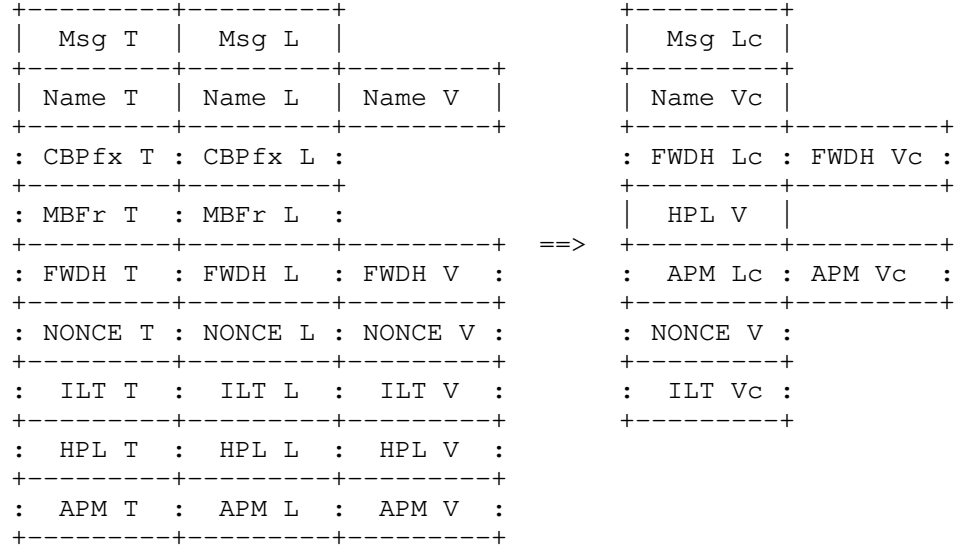


Figure 12: Compression of NDN LoWPAN Interest Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 13.

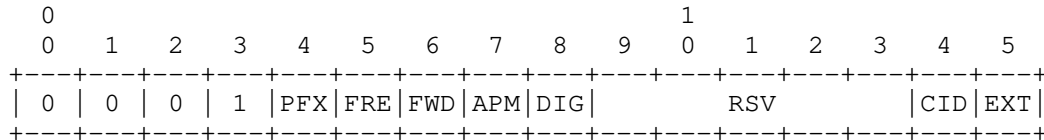


Figure 13: Dispatch format for compressed NDN Interest messages

PFX: CanBePrefix TLV 0: The uncompressed message does not include a CanBePrefix TLV.

1: The uncompressed message does include a CanBePrefix TLV and is removed from the compressed message.

FRE: MustBeFresh TLV 0: The uncompressed message does not include a MustBeFresh TLV.

1: The uncompressed message does include a

MustBeFresh TLV and is removed from the compressed message.

FWD: ForwardingHint TLV 0: The uncompressed message does not include a ForwardingHint TLV.

1: The uncompressed message does include a ForwardingHint TLV. The Type field is removed from the compressed message. Further, all link delegation types and link preference types are removed. All included names are compressed according to Section 5.2. If any name is not compressible, the message MUST be sent uncompressed.

APM: ApplicationParameters TLV 0: The uncompressed message does not include an ApplicationParameters TLV.

1: The uncompressed message does include an ApplicationParameters TLV. The Type field is removed from the compressed message.

DIG: ImplicitSha256DigestComponent TLV 0: The name does not include an ImplicitSha256DigestComponent as the last TLV.

1: The name does include an ImplicitSha256DigestComponent as the last TLV. The Type and Length fields are omitted.

RSV: Reserved Must be set to 0.

CID: Context Identifier See Figure 5.

EXT: Extension 0: No extension byte follows.

1: Extension byte EXT_0 follows immediately. See Section 5.3.3.

5.3.3. Dispatch Extension

The EXT_0 byte follows the description in Section 4.1.1 and is illustrated in Figure 14.

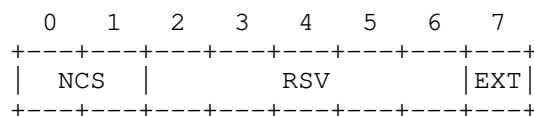


Figure 14: EXT_0 format

NCS: Name Compression Strategy 00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension 0: No extension byte follows.

1: A further extension byte follows immediately.

5.4. Data Messages

5.4.1. Uncompressed Data Messages

An uncompressed Data message uses the base dispatch format and sets the C flag to 0, the P flag to 0 and the M flag to 1 (Figure 15). The Data message is handed to the NDN network stack without modifications.

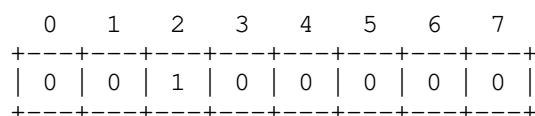


Figure 15: Dispatch format for uncompressed NDN Data messages

5.4.2. Compressed Data Messages

The compressed Data message uses the extended dispatch format (Figure 5) and sets the C as well as the M flags to 1. The P flag is set to 0. If a Data message contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

By default, the Data message is compressed with the following base rule set:

1. The Type field of the outermost MessageType TLV is removed.
2. The Name TLV is compressed according to Section 5.2. For this, all NameComponents are expected to be of type GenericNameComponent and to have a length greater than 0. In any other case, the message MUST be sent uncompressed.

3. The MetaInfo TLV Type and Length fields are elided from the compressed Data message.
4. The FreshnessPeriod TLV MUST be moved to the end of the compressed Data message. Type and Length fields are elided and the value is encoded as described in Section 7 as a 1-byte time-code. If the freshness period is not a valid time-value, then the message MUST be sent uncompressed in order to preserve the security envelope of the Data message. The presence of a FreshnessPeriod TLV is deduced from the remaining one byte length to parse.
5. The Type fields of the SignatureInfo TLV, SignatureType TLV and SignatureValue TLV are removed.

The compressed NDN LoWPAN Data message is visualized in Figure 16.

T = Type, L = Length, V = Value

Lc = Compressed Length, Vc = Compressed Value

: = optional field, | = mandatory field

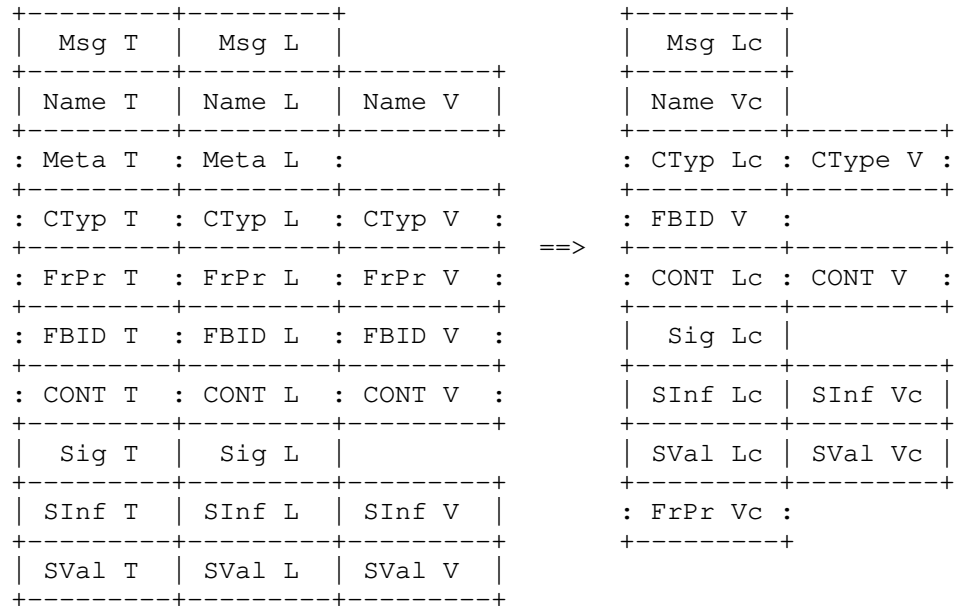


Figure 16: Compression of NDN LoWPAN Data Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 17.

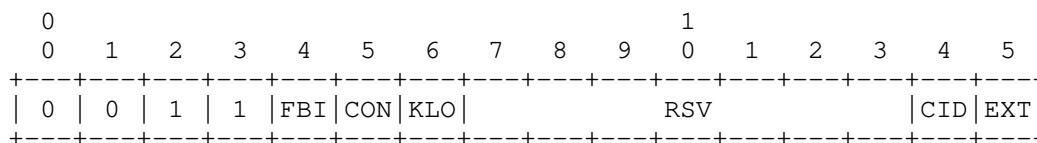


Figure 17: Dispatch format for compressed NDN Data messages

FBI: FinalBlockId TLV 0: The uncompressed message does not include a FinalBlockId TLV.

1: The uncompressed message does include a FinalBlockId and it is encoded according to Section 5.2. If the FinalBlockId TLV is not compressible, then the message MUST be sent uncompressed.

CON: ContentType TLV 0: The uncompressed message does not include a ContentType TLV.

1: The uncompressed message does include a ContentType TLV. The Type field is removed from the compressed message.

KLO: KeyLocator TLV 0: If the included SignatureType requires a KeyLocator TLV, then the KeyLocator represents a name and is compressed according to Section 5.2. If the name is not compressible, then the message MUST be sent uncompressed.

1: If the included SignatureType requires a KeyLocator TLV, then the KeyLocator represents a KeyDigest. The Type field of this KeyDigest is removed.

RSV: Reserved Must be set to 0.

CID: Context Identifier See Figure 5.

EXT: Extension 0: No extension byte follows.

1: Extension byte EXT_0 follows immediately. See Section 5.4.3.

5.4.3. Dispatch Extension

The EXT_0 byte follows the description in Section 4.1.1 and is illustrated in Figure 18.

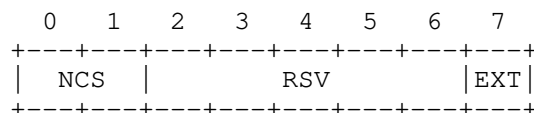


Figure 18: EXT_0 format

NCS: Name Compression Strategy 00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension 0: No extension byte follows.

1: A further extension byte follows immediately.

6. Space-efficient Message Encoding for CCNx

6.1. TLV Encoding

The generic CCNx TLV encoding is described in [RFC8609]. Type and Length fields attain the common fixed length of 2 bytes.

The TLV encoding for CCNx LoWPAN is changed to the more space efficient encoding described in Section 5.1. Hence NDN and CCNx use the same compressed format for writing TLVs.

6.2. Name TLV Compression

Name TLVs are compressed using the scheme already defined in Section 5.2 for NDN. If a Name TLV contains T_IPID, T_APP, or organizational TLVs, then the name remains uncompressed.

6.3. Interest Messages

6.3.1. Uncompressed Interest Messages

An uncompressed Interest message uses the base dispatch format (see Figure 4) and sets the C as well as the M flag to 0. The P flag is set to 1 (Figure 19). The Interest message is handed to the CCNx network stack without modifications.

0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0

Figure 19: Dispatch format for uncompressed CCNx Interest messages

6.3.2. Compressed Interest Messages

The compressed Interest message uses the extended dispatch format (Figure 5) and sets the C and P flags to 1. The M flag is set to 0. If an Interest message contains TLVs that are not mentioned in the following compression rules, then this message **MUST** be sent uncompressed.

In the default use case, the Interest message is compressed with the following minimal rule set:

1. The version is elided from the Fixed Header and assumed to be 1.
2. The Type and Length fields of the CCNx Message TLV are elided and are obtained from the Fixed Header on decompression.

The compressed CCNx LoWPAN Interest message is visualized in Figure 20.

T = Type, L = Length, V = Value
 Lc = Compressed Length, Vc = Compressed Value
 : = optional field, | = mandatory field

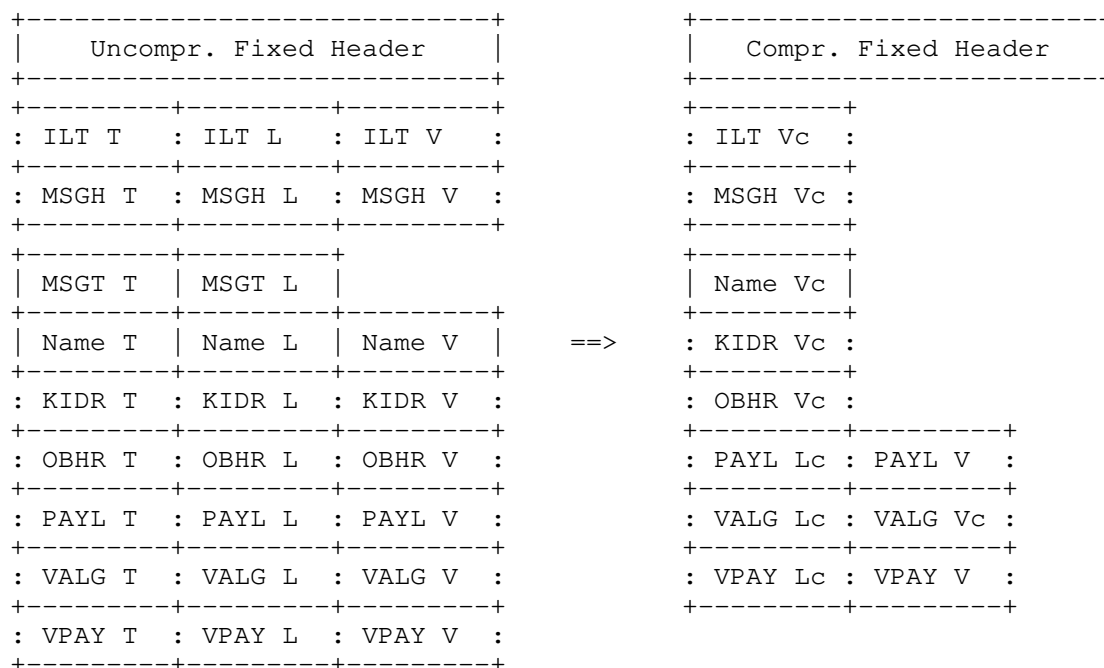


Figure 20: Compression of CCNx LoWPAN Interest Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 21.

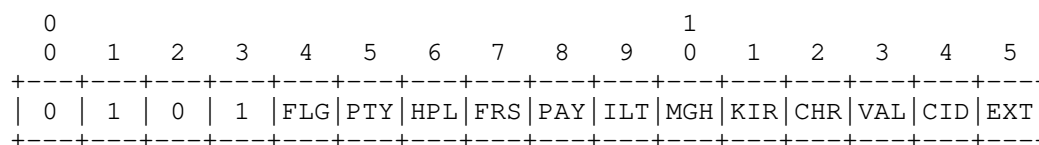


Figure 21: Dispatch format for compressed CCNx Interest messages

FLG: Flags field in the fixed header 0: The Flags field equals 0 and is removed from the Interest message.

1: The Flags field appears in the fixed header.

PTY: PacketType field in the fixed header 0: The PacketType field

is elided and assumed to be PT_INTEREST.

1: The PacketType field
is elided and assumed to be PT_RETURN.

HPL: HopLimit field in the fixed header 0: The HopLimit field
appears in the fixed header.

1: The HopLimit field is
elided and assumed to be 1.

FRS: Reserved field in the fixed header 0: The Reserved field
appears in the fixed header.

1: The Reserved field is
elided and assumed to be 0.

PAY: Optional Payload TLV 0: The Payload TLV is absent.

1: The Payload TLV is present and the
type field is elided.

ILT: Optional Hop-By-Hop InterestLifetime TLV See Section 6.3.2.1 for further
details on the ordering
of hop-by-hop TLVs.

0: No
InterestLifetime TLV is present in the Interest message.

1: An
InterestLifetime TLV is present with a fixed length of 1
byte and is encoded as described in Section 7. The type
and length fields are elided.

MGH: Optional Hop-By-Hop MessageHash TLV See Section 6.3.2.1 for further deta
ils on the ordering
of hop-by-hop TLVs.

This TLV is expected to contain a T_SHA-256 TLV. If
another hash is contained, then the Interest MUST be sent
uncompressed.

0: The MessageHash TLV is
absent.

1: A T_SHA-256 TLV is
present and the type as well as the length fields are
removed. The length field is assumed to represent 32
bytes. The outer Message Hash TLV is omitted.

KIR: Optional KeyIdRestriction TLV This TLV is expected to contain a T_SHA-256 TLV. If

another hash is contained, then the Interest MUST be sent uncompressed.

0: The KeyIdRestriction TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer KeyIdRestriction TLV is omitted.

CHR: Optional ContentObjectHashRestriction TLV This TLV is expected to contain a T_SHA-256 TLV. If

another hash is contained, then the Interest MUST be sent uncompressed.

0: The ContentObjectHashRestriction TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer ContentObjectHashRestriction TLV is omitted.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs 0: No validation related TLVs are present in the Interest message.

1: Validation related TLVs are present in the Interest message. An additional byte follows immediately that handles validation related TLV compressions and is described in Section 6.3.2.2.

CID: Context Identifier See Figure 5.

EXT: Extension 0: No extension byte follows.

1: Extension byte EXT_0 follows immediately. See Section 6.3.3.

6.3.2.1. Hop-By-Hop Header TLVs Compression

Hop-By-Hop Header TLVs are unordered. For an Interest message, two optional Hop-By-Hop Header TLVs are defined in [RFC8609], but several more can be defined in higher level specifications. For the compression specified in the previous section, the Hop-By-Hop TLVs are ordered as follows:

1. Interest Lifetime TLV
2. Message Hash TLV

Note: Other Hop-By-Hop Header TLVs than those two remain uncompressed in the encoded message and they appear in the same order as in the original message, but after the Interest Lifetime TLV and Message Hash TLV.

6.3.2.2. Validation

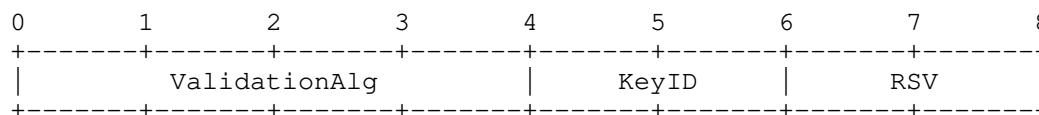


Figure 22: Dispatch for Interest Validations

ValidationAlg: Optional ValidationAlgorithm TLV 0000: An uncompressed ValidationAlgorithm TLV is included.

0001: A T_CRC32C ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included.

0010: A T_CRC32C ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included. Additionally, a Sigtime TLV is inlined without a type and a length field.

0011: A T_HMAC-SHA256 ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included.

0100: A T_HMAC-SHA256 ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included. Additionally, a Sigtime TLV is inlined without a type and a length field.

0101: Reserved.

0110: Reserved.

0111: Reserved.

1000: Reserved.

1001: Reserved.

1010: Reserved.

1011: Reserved.

1100: Reserved.

1101: Reserved.

1110: Reserved.

1111: Reserved.

KeyID: Optional KeyID TLV within the ValidationAlgorithm TLV 00: The KeyId TLV is absent.

01: The KeyId TLV is present and uncompressed.

10: A T_SHA-256 TLV is present and the type field as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer KeyId TLV is omitted.

11: A T_SHA-512 TLV is present and the type field as well as the length fields are removed. The length field is assumed to represent 64 bytes. The outer KeyId TLV is omitted.

RSV: Reserved Must be set to 0.

The ValidationPayload TLV is present if the ValidationAlgorithm TLV is present. The type field is omitted.

6.3.3. Dispatch Extension

The EXT_0 byte follows the description in Section 4.1.1 and is illustrated in Figure 23.

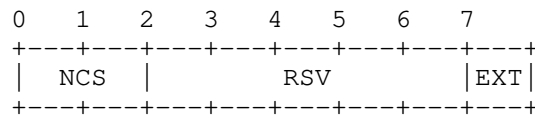


Figure 23: EXT_0 format

NCS: Name Compression Strategy 00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension 0: No extension byte follows.

1: A further extension byte follows immediately.

6.4. Content Objects

6.4.1. Uncompressed Content Objects

An uncompressed Content object uses the base dispatch format (see Figure 4) and sets the C flag to 0, the P and M flags to 1 (Figure 24). The Content object is handed to the CCNx network stack without modifications.

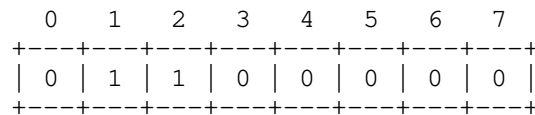


Figure 24: Dispatch format for uncompressed CCNx Content objects

6.4.2. Compressed Content Objects

The compressed Content object uses the extended dispatch format (Figure 5) and sets the C, P, as well as the M flag to 1. If a Content object contains TLVs that are not mentioned in the following compression rules, then this message MUST be sent uncompressed.

By default, the Content object is compressed with the following base rule set:

1. The version is elided from the Fixed Header and assumed to be 1.
2. The PacketType field is elided from the Fixed Header.
3. The Type and Length fields of the CCNx Message TLV are elided and are obtained from the Fixed Header on decompression.

The compressed CCNx LoWPAN Data message is visualized in Figure 25.

T = Type, L = Length, V = Value

Lc = Compressed Length, Vc = Compressed Value

: = optional field, | = mandatory field

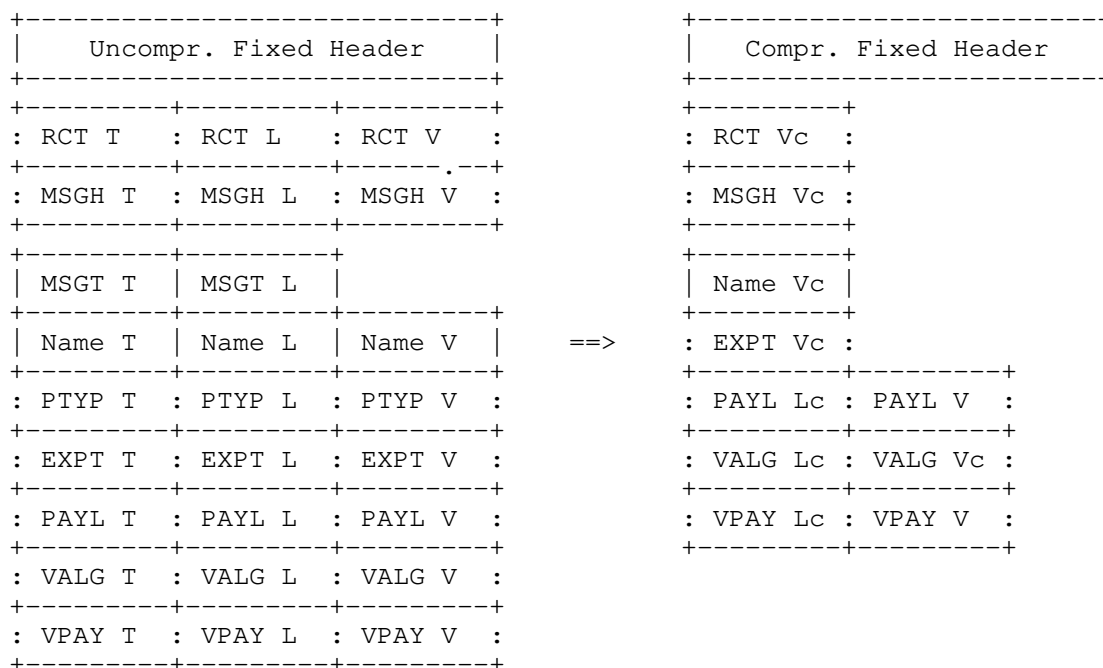


Figure 25: Compression of CCNx LoWPAN Data Message

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 26.

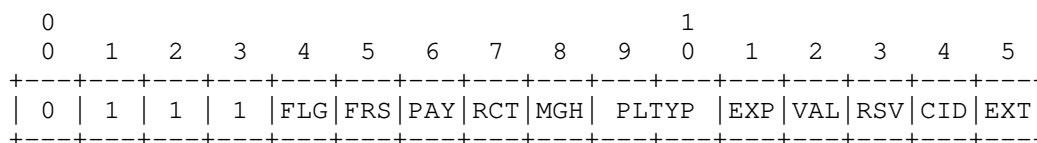


Figure 26: Dispatch format for compressed CCNx Content objects

FLG: Flags field in the fixed header See Section 6.3.2.

FRS: Reserved field in the fixed header See Section 6.3.2.

PAY: Optional Payload TLV See Section 6.3.2.

RCT: Optional Hop-By-Hop RecommendedCacheTime TLV 0: The Recommended Cache Time TLV is absent.

1: The Recommended Cache Time TLV is present and the type as well as the length fields are elided.

MGH: Optional Hop-By-Hop MessageHash TLV See Section 6.4.2.1 for further details on the ordering of hop-by-hop TLVs.

This TLV is expected to contain a T_SHA-256 TLV. If another hash is contained, then the Content Object MUST be sent uncompressed.

0: The MessageHash TLV is absent.

1: A T_SHA-256 TLV is present and the type as well as the length fields are removed. The length field is assumed to represent 32 bytes. The outer Message Hash TLV is omitted.

PLTYP: Optional PayloadType TLV 00: The PayloadType TLV is absent.

01: The PayloadType TLV is absent and T_PAYLOADTYPE_DATA is assumed.

10: The PayloadType TLV is absent and T_PAYLOADTYPE_KEY is assumed.

11: The PayloadType TLV is present and uncompressed.

EXP: Optional ExpiryTime TLV 0: The ExpiryTime TLV is absent.

1: The ExpiryTime TLV is present and the type as well as the length fields are elided.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs See Sec

tion 6.3.2.

RSV: Reserved Must be set to 0.

CID: Context Identifier See Figure 5.

EXT: Extension 0: No extension byte follows.

1: Extension byte EXT_0 follows immediately. See Section 6.4.3.

6.4.2.1. Hop-By-Hop Header TLVs Compression

Hop-By-Hop Header TLVs are unordered. For a Content Object message, two optional Hop-By-Hop Header TLVs are defined in [RFC8609], but several more can be defined in higher level specifications. For the compression specified in the previous section, the Hop-By-Hop TLVs are ordered as follows:

1. Recommended Cache Time TLV
2. Message Hash TLV

Note: Other Hop-By-Hop Header TLVs than those two remain uncompressed in the encoded message and they appear in the same order as in the original message, but after the Recommended Cache Time TLV and Message Hash TLV.

6.4.3. Dispatch Extension

The EXT_0 byte follows the description in Section 4.1.1 and is illustrated in Figure 27.

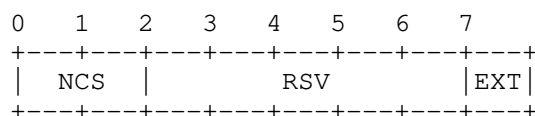


Figure 27: EXT_0 format

NCS: Name Compression Strategy 00: Names are compressed with the default name compression strategy (see Section 5.2).

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved Must be set to 0.

EXT: Extension 0: No extension byte follows.

1: A further extension byte follows immediately.

7. Compressed Time Encoding

This document adopts the 8-bit compact time representation for relative time values described in Section 5 of [RFC5497] with the constant factor C set to $C := 1/32$.

Valid time offsets in CCNx and NDN reach from a few milliseconds (e.g., lifetime of low-latency Interests) to several years (e.g., content freshness periods in caches). Therefore, this document adds two modifications to the compression algorithm.

The first modification is the inclusion of a subnormal form [IEEE.754.2019] for time-codes with exponent 0 to provide an increased precision and a gradual underflow for the smallest numbers. The formula is changed as follows (a := mantissa; b := exponent):

Subnormal (b == 0): $(0 + a/8) * 2 * C$

Normalized (b > 0): $(1 + a/8) * 2^b * C$ (see [RFC5497])

This configuration allows for the following ranges:

- * Minimum subnormal number: 0 seconds
- * 2nd minimum subnormal number: ~0.007812 seconds
- * Maximum subnormal number: ~0.054688 seconds
- * Minimum normalized number: ~0.062500 seconds
- * 2nd minimum normalized number: ~0.070312 seconds
- * Maximum normalized number: ~3.99 years

The second modification only applies to uncompressible time offsets that are outside any security envelope. An invalid time-value MUST be set to the largest valid time-value that is smaller than the invalid input value before compression.

8. Stateful Header Compression

Stateful header compression in ICN LoWPAN enables packet size reductions in two ways. First, common information that is shared throughout the local LoWPAN may be memorized in context state at all nodes and omitted from communication. Second, redundancy in a single Interest-data exchange may be removed from ICN stateful forwarding on a hop-by-hop bases and memorized in en-route state tables.

8.1. LoWPAN-local State

A context identifier (CID) is a byte that refers to a particular conceptual context between network devices and MAY be used to replace frequently appearing information, such as name prefixes, suffixes, or meta information, such as Interest lifetime.

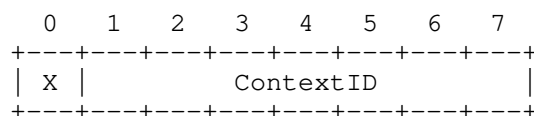


Figure 28: Context Identifier.

The 7-bit ContextID is a locally-scoped unique identifier that represents contextual state shared between sender and receiver of the corresponding frame (see Figure 28). If set the most significant bit indicates the presence of another, subsequent ContextID byte (see Figure 33).

Context state shared between senders and receivers is removed from the compressed packet prior to sending, and reinserted after reception prior to passing to the upper stack.

The actual information in a context and how it is encoded are out of scope of this document. The initial distribution and maintenance of shared context is out of scope of this document. Frames containing unknown or invalid CIDs MUST be silently discarded.

8.2. En-route State

In CCNx and NDN, Name TLVs are included in Interest messages, and they return in data messages. Returning Name TLVs either equal the original Name TLV, or they contain the original Name TLV as a prefix. ICN LoWPAN reduces this redundancy in responses by replacing Name TLVs with single bytes that represent link-local HopIDs. HopIDs are carried as Context Identifiers (see Section 8.1) of link-local scope as shown in Figure 29.

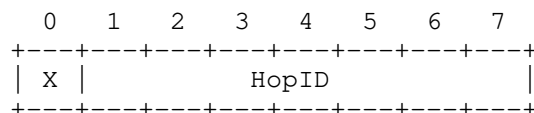


Figure 29: Context Identifier as HopID.

A HopID is valid if not all ID bits are set to zero and invalid otherwise. This yields 127 distinct HopIDs. If this range (1...127) is exhausted, the messages MUST be sent without en-route state compression until new HopIDs are available. An ICN LoWPAN node that forwards without replacing the name by a HopID (without en-route compression) MUST invalidate the HopID by setting all ID-bits to zero.

While an Interest is traversing, a forwarder generates an ephemeral HopID that is tied to a PIT entry. Each HopID MUST be unique within the local PIT and only exists during the lifetime of a PIT entry. To maintain HopIDs, the local PIT is extended by two new columns: HIDi (inbound HopIDs) and HIDo (outbound HopIDs).

HopIDs are included in Interests and stored on the next hop with the resulting PIT entry in the HIDi column. The HopID is replaced with a newly generated local HopID before the Interest is forwarded. This new HopID is stored in the HIDo column of the local PIT (see Figure 30).

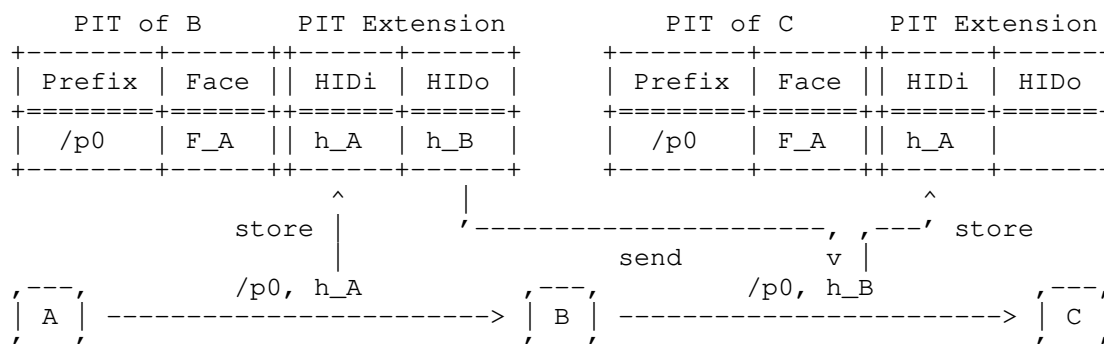


Figure 30: Setting compression state en-route (Interest).

Responses include HopIDs that were obtained from Interests. If the returning Name TLV equals the original Name TLV, then the name is entirely elided. Otherwise, only the matching name prefix is elided and the distinct name suffix is included along with the HopID. When a response is forwarded, the contained HopID is extracted and used to match against the correct PIT entry by performing a lookup on the HIDo column. The HopID is then replaced with the corresponding HopID from the HIDi column prior to forwarding the response (Figure 31).

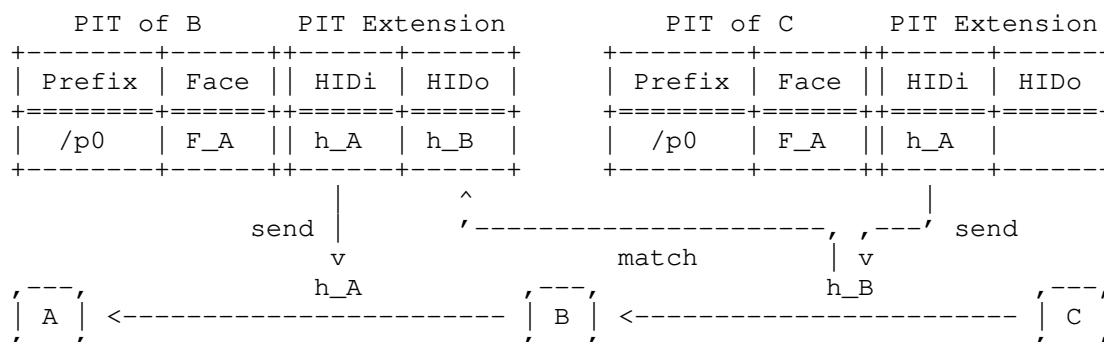


Figure 31: Eliding Name TLVs using en-route state (data).

It should be noted that each forwarder of an Interest in an ICN LoWPAN network can individually decide whether to participate in en-route compression or not. However, an ICN LoWPAN node SHOULD use en-route compression whenever the stateful compression mechanism is activated.

Note also that the extensions of the PIT data structure are required only at ICN LoWPAN nodes, while regular NDN/CCNx forwarders outside of an ICN LoWPAN domain do not need to implement these extensions.

8.3. Integrating Stateful Header Compression

A CID appears whenever the CID flag is set (see Figure 5). The CID is appended to the last ICN LoWPAN dispatch byte as shown in Figure 32.

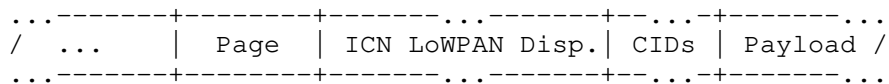


Figure 32: LoWPAN Encapsulation with ICN LoWPAN and CIDs

Multiple CIDs are chained together, with the most significant bit indicating the presence of a subsequent CID (Figure 33). This allows to use multiple shared contexts in compressed messages.

The HopID is always included as the very first CID.

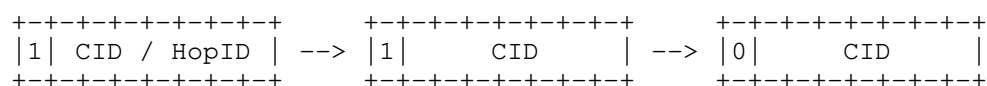


Figure 33: Chaining of context identifiers.

9. ICN LoWPAN Constants and Variables

This is a summary of all ICN LoWPAN constants and variables.

DEFAULT_NDN_HOPLIMIT: 255

10. Implementation Report and Guidance

The ICN LoWPAN scheme defined in this document has been implemented as an extension of the NDN/CCNx software stack [CCN-LITE] in its IoT version on RIOT [RIOT]. An experimental evaluation for NDN over ICN LoWPAN with varying configurations has been performed in [ICNLOWPAN]. Energy profilings and processing time measurements indicate significant energy savings, while amortized costs for processing show no penalties.

10.1. Preferred Configuration

The header compression performance depends on certain aspects and configurations. It works best for the following cases:

- * Signed time offsets compress as per Section 7 without the need for rounding.
- * Contextual state (e.g., prefixes) is distributed, such that long names can be elided from Interest and data messages.
- * Frequently used TLV type numbers for CCNx and NDN stay in the lower range (< 255).

Name components are of GenericNameComponent type and are limited to a length of 15 bytes to enable compression for all messages.

10.2. Further Experimental Deployments

An investigation of ICN LoWPAN in large-scale deployments with varying traffic patterns using larger samples of the different board types available remains as future work. This document will be revised to progress it to the Standards Track, once sufficient operational experience has been acquired. Experience reports are encouraged, particularly in the following areas:

- * The name compression scheme (Section 5.2) is optimized for short name components of GenericNameComponent type. An empirical study on name lengths in different deployments of selected use cases, such as smart home, smart city, and industrial IoT can provide meaningful reports on necessary name component types and lengths. A conclusive outcome helps to understand whether and how extension mechanisms are needed (Section 5.3.3). As a preliminary analysis, [ICNLOWPAN] investigates the effectiveness of the proposed compression scheme with URLs obtained from the WWW. Studies on CoAP [RFC7252] deployments can offer additional insights on naming schemes in the IoT.
- * The fragmentation scheme (Section 4.2) inherited from 6LoWPAN allows for a transparent, hop-wise reassembly of CCNx or NDN packets. Fragment forwarding [RFC8930] with selective fragment recovery [RFC8931] can improve the end-to-end latency and reliability, while it reduces buffer requirements on forwarders. Initial evaluations ([SFR-ICNLOWPAN]) show that a naive integration of these upcoming fragmentation features into ICN LoWPAN renders the hop-wise content replication inoperative, since Interest and data messages are reassembled end-to-end. More deployment experiences are necessary to gauge the feasibility of different fragmentation schemes in ICN LoWPAN.
- * Context state (Section 8.1) holds information that is shared between a set of devices in a LoWPAN. Fixed name prefixes and suffixes are good candidates to be distributed to all nodes in order to elide them from request and response messages. More experience and a deeper inspection of currently available and upcoming protocol features is necessary to identify other protocol fields.
- * The distribution and synchronization of contextual state can potentially be adopted from Section 7.2 of [RFC6775], but requires further evaluations. While 6LoWPAN uses the Neighbor Discovery protocol to disseminate state, CCNx and NDN deployments are missing out on a standard mechanism to bootstrap and manage configurations.

- * The stateful en-route compression (Section 8.2) supports a limited number of 127 distinct HopIDs that can be simultaneously in use on a single node. Complex deployment scenarios that make use of multiple, concurrent requests can provide a better insight on the number of open requests stored in the Pending Interest Table of memory-constrained devices. This number can serve as an upper-bound and determines whether the HopID length needs to be resized to fit more HopIDs to the cost of additional header overhead.
- * Multiple implementations that generate and deploy the compression options of this memo in different ways will also add to the experience and understanding of the benefits and limitations of the proposed schemes. Different reports can help to illuminate on the complexity of implementing ICN LoWPAN for constrained devices, as well as on maintaining interoperability with other implementations.

11. Security Considerations

Main memory is typically a scarce resource of constrained networked devices. Fragmentation as described in this memo preserves fragments and purges them only after a packet is reassembled, which requires a buffering of all fragments. This scheme is able to handle fragments for distinctive packets simultaneously, which can lead to overflowing packet buffers that cannot hold all necessary fragments for packet reassembly. Implementers are thus urged to make use of appropriate buffer replacement strategies for fragments. Minimal fragment forwarding [RFC8930] can potentially prevent fragment buffer saturation in forwarders.

The stateful header compression generates ephemeral HopIDs for incoming and outgoing Interests and consumes them on returning Data packets. Forged Interests can deplete the number of available HopIDs, thus leading to a denial of compression service for subsequent content requests.

To further alleviate the problems caused by forged fragments or Interest initiations, proper protective mechanisms for accessing the link-layer should be deployed. IEEE 802.15.4, e.g., provides capabilities to protect frames and restrict them to a point-to-point link, or a group of devices.

12. IANA Considerations

12.1. Reserving Space in the 6LoWPAN Dispatch Type Field Registry

IANA has assigned dispatch values of the 6LoWPAN Dispatch Type Field registry [RFC4944][RFC8025] with Page TBD1 for ICN LoWPAN. Table 2 represents updates to the registry.

Bit Pattern	Page	Header Type
00 000000	TBD1	Uncompressed NDN Interest messages
00 01xxxx	TBD1	Compressed NDN Interest messages
00 100000	TBD1	Uncompressed NDN Data messages
00 11xxxx	TBD1	Compressed NDN Data messages
01 000000	TBD1	Uncompressed CCNx Interest messages
01 01xxxx	TBD1	Compressed CCNx Interest messages
01 100000	TBD1	Uncompressed CCNx Content Object messages
01 11xxxx	TBD1	Compressed CCNx Content Object messages

Table 2: Dispatch types for NDN and CCNx with page TBD1.

13. References

13.1. Normative References

[IEEE.754.2019]

Institute of Electrical and Electronics Engineers, C/MS C - Microprocessor Standards Committee, "Standard for Floating-Point Arithmetic", June 2019, <<https://standards.ieee.org/content/ieee-standards/en/standard/754-2019.html>>.

[ieee802.15.4]

"IEEE Std. 802.15.4-2015", April 2016, <<https://standards.ieee.org/findstds/standard/802.15.4-2015.html>>.

[RFC2119]

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

- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, DOI 10.17487/RFC5497, March 2009, <<https://www.rfc-editor.org/info/rfc5497>>.
- [RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, DOI 10.17487/RFC6256, May 2011, <<https://www.rfc-editor.org/info/rfc6256>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.

13.2. Informative References

- [CCN-LITE] "CCN-lite: A lightweight CCNx and NDN implementation", <<http://ccn-lite.net/>>.
- [I-D.irtf-icnrg-flic] Tschudin, C., Wood, C., Mosko, M., and D. Oran, "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-02, 4 November 2019, <<http://www.ietf.org/internet-drafts/draft-irtf-icnrg-flic-02.txt>>.
- [ICNLOWPAN] Gundogan, C., Kietzmann, P., Schmidt, TC., and M. Waehlich, "ICNLoWPAN -- Named-Data Networking in Low Power IoT Networks", Proc. of 18th IFIP Networking Conference, May 2019, <<https://doi.org/10.23919/IFIPNetworking.2019.8816850>>.
- [NDN] Jacobson, V., Smetters, D., Thornton, J., and M. Plass, "Networking Named Content", 5th Int. Conf. on emerging Networking Experiments and Technologies (ACM CoNEXT), 2009, <<https://doi.org/10.1145/1658939.1658941>>.

- [NDN-EXP1] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, TC., and M. Waehlich, "Information Centric Networking in the IoT: Experiments with NDN in the Wild", Proc. of 1st ACM Conf. on Information-Centric Networking (ICN-2014) ACM DL, pp. 77-86, September 2014, <<http://dx.doi.org/10.1145/2660129.2660144>>.
- [NDN-EXP2] Gundogan, C., Kietzmann, P., Lenders, M., Petersen, H., Schmidt, TC., and M. Waehlich, "NDN, CoAP, and MQTT: A Comparative Measurement Study in the IoT", Proc. of 5th ACM Conf. on Information-Centric Networking (ICN-2018) ACM DL, pp. 159-171, September 2018, <<https://doi.org/10.1145/3267955.3267967>>.
- [NDN-MAC] Kietzmann, P., Gundogan, C., Schmidt, TC., Hahm, O., and M. Waehlich, "The Need for a Name to MAC Address Mapping in NDN: Towards Quantifying the Resource Gain", Proc. of 4th ACM Conf. on Information-Centric Networking (ICN-2017) ACM DL, pp. 36-42, September 2017, <<https://doi.org/10.1145/3125719.3125737>>.
- [NDN-PACKET-SPEC] "NDN Packet Format Specification", <<https://named-data.net/doc/NDN-packet-spec/0.3/>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [RFC7476] Pentikousis, K., Ed., Ohlman, B., Corujo, D., Boggia, G., Tyson, G., Davies, E., Molinaro, A., and S. Eum, "Information-Centric Networking: Baseline Scenarios", RFC 7476, DOI 10.17487/RFC7476, March 2015, <<https://www.rfc-editor.org/info/rfc7476>>.
- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.

- [RFC7945] Pentikousis, K., Ed., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", RFC 7945, DOI 10.17487/RFC7945, September 2016, <<https://www.rfc-editor.org/info/rfc7945>>.
- [RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <<https://www.rfc-editor.org/info/rfc8025>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.
- [RFC8930] Watteyne, T., Ed., Thubert, P., Ed., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multi-Hop IPv6 Network", RFC 8930, DOI 10.17487/RFC8930, November 2020, <<https://www.rfc-editor.org/info/rfc8930>>.
- [RFC8931] Thubert, P., Ed., "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery", RFC 8931, DOI 10.17487/RFC8931, November 2020, <<https://www.rfc-editor.org/info/rfc8931>>.
- [RIOT] Baccelli, E., Gundogan, C., Hahm, O., Kietzmann, P., Lenders, MS., Petersen, H., Schleiser, K., Schmidt, TC., and M. Waehlich, "RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT", IEEE Internet of Things Journal Vol. 5, No. 6, p. 4428-4440, December 2018, <<https://doi.org/10.1109/JIOT.2018.2815038>>.
- [SFR-ICNLOWPAN] Lenders, M., Gundogan, C., Schmidt, TC., and M. Waehlich, "Connecting the Dots: Selective Fragment Recovery in ICNLoWPAN", Proc. of 7th ACM Conf. on Information-Centric Networking (ICN-2020) ACM DL, pp. 70-76, September 2020, <<https://doi.org/10.1145/3405656.3418719>>.

[TLV-ENC-802.15.4]
"CCN and NDN TLV encodings in 802.15.4 packets",
<<https://datatracker.ietf.org/meeting/interim-2015-icnrg-01/materials/slides-interim-2015-icnrg-1-2>>.

[WIRE-FORMAT-CONSID]
"CCN/NDN Protocol Wire Format and Functionality
Considerations", <<https://datatracker.ietf.org/meeting/interim-2015-icnrg-01/materials/slides-interim-2015-icnrg-1-8>>.

Appendix A. Estimated Size Reduction

In the following a theoretical evaluation is given to estimate the gains of ICN LoWPAN compared to uncompressed CCNx and NDN messages.

We assume that n is the number of name components, $comps_n$ denotes the sum of n name component lengths. We also assume that the length of each name component is lower than 16 bytes. The length of the content is given by $clen$. The lengths of TLV components is specific to the CCNx or NDN encoding and outlined below.

A.1. NDN

The NDN TLV encoding has variable-sized TLV fields. For simplicity, the 1 byte form of each TLV component is assumed. A typical TLV component therefore is of size 2 (type field + length field) + the actual value.

A.1.1. Interest

Figure 34 depicts the size requirements for a basic, uncompressed NDN Interest containing a CanBePrefix TLV, a MustBeFresh TLV, a InterestLifetime TLV set to 4 seconds and a HopLimit TLV set to 6. Numbers below represent the amount of bytes.

Interest TLV	= 2	= 21 + 2n + comps_n
Name	2 +	
NameComponents	= 2n +	
	comps_n	
CanBePrefix	= 2	
MustBeFresh	= 2	
Nonce	= 6	
InterestLifetime	= 4	
HopLimit	= 3	

Figure 34: Estimated size of an uncompressed NDN Interest

Figure 35 depicts the size requirements after compression.

Dispatch Page Switch	= 1	= 10 + n/2 + comps_n
NDN Interest Dispatch	= 2	
Interest TLV	= 1	
Name		
NameComponents	= n/2 +	
	comps_n	
Nonce	= 4	
HopLimit	= 1	
InterestLifetime	= 1	

Figure 35: Estimated size of a compressed NDN Interest

The size difference is: 11 + 1.5n bytes.

For the name /DE/HH/HAW/BT7, the total size gain is 17 bytes, which is 43% of the uncompressed packet.

A.1.1.2. Data

Figure 36 depicts the size requirements for a basic, uncompressed NDN Data containing a FreshnessPeriod as MetaInfo. A FreshnessPeriod of 1 minute is assumed and the value is encoded using 1 byte. An HMACWithSha256 is assumed as signature. The key locator is assumed to contain a Name TLV of length klen.

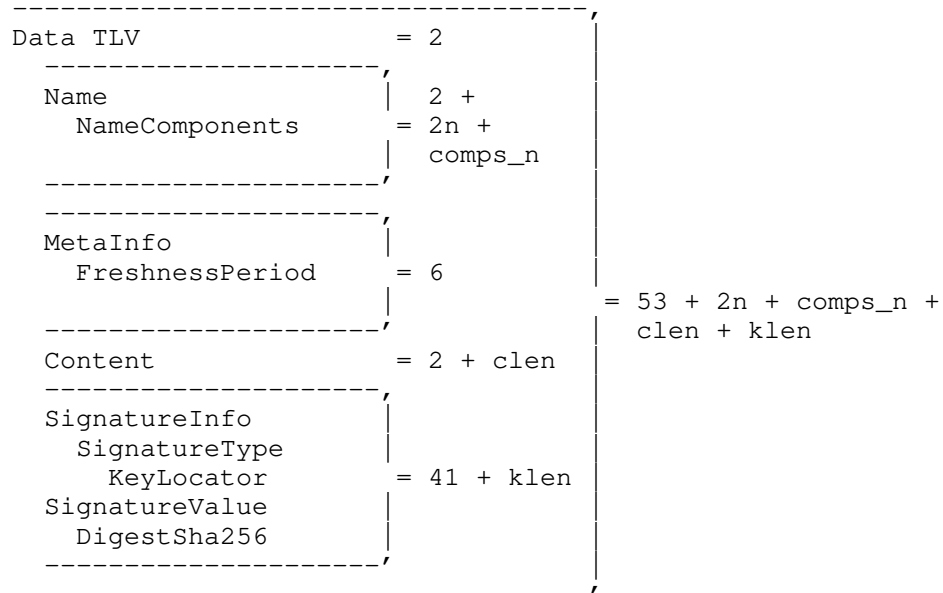


Figure 36: Estimated size of an uncompressed NDN Data

Figure 37 depicts the size requirements for the compressed version of the above Data packet.

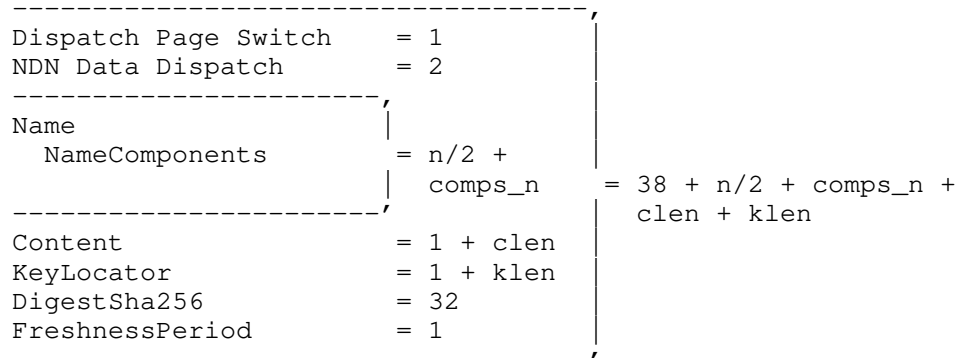


Figure 37: Estimated size of a compressed NDN Data

The size difference is: $15 + 1.5n$ bytes.

For the name /DE/HH/HAW/BT7, the total size gain is 21 bytes.

A.2. CCNx

The CCNx TLV encoding defines a 2-byte encoding for type and length fields, summing up to 4 bytes in total without a value.

A.2.1. Interest

Figure 38 depicts the size requirements for a basic, uncompressed CCNx Interest. No Hop-By-Hop TLVs are included, the protocol version is assumed to be 1 and the reserved field is assumed to be 0. A KeyIdRestriction TLV with T_SHA-256 is included to limit the responses to Content Objects containing the specific key.

Fixed Header	= 8	}	= 56 + 4n + comps_n
Message	= 4		
Name	4 +		
NameSegments	= 4n +		
	comps_n		
KeyIdRestriction	= 40		

Figure 38: Estimated size of an uncompressed CCNx Interest

Figure 39 depicts the size requirements after compression.

Dispatch Page Switch	= 1	}	= 38 + n/2 + comps_n
CCNx Interest Dispatch	= 2		
Fixed Header	= 3		
Name			
NameSegments	= n/2 +		
	comps_n		
T_SHA-256	= 32		

Figure 39: Estimated size of a compressed CCNx Interest

The size difference is: $18 + 3.5n$ bytes.

For the name /DE/HH/HAW/BT7, the size is reduced by 53 bytes, which is 53% of the uncompressed packet.

A.2.2. Content Object

Figure 40 depicts the size requirements for a basic, uncompressed CCNx Content Object containing an ExpiryTime Message TLV, an HMAC_SHA-256 signature, the signature time and a hash of the shared secret key. In the fixed header, the protocol version is assumed to be 1 and the reserved field is assumed to be 0

Fixed Header	= 8	
Message	= 4	
Name	4 +	
NameSegments	= 4n +	
	comps_n	
ExpiryTime	= 12	
Payload	= 4 + clen	
ValidationAlgorithm		
T_HMAC-256	= 56	
KeyId		
SignatureTime		
ValidationPayload	= 36	
		= 124 + 4n + comps_n + clen

Figure 40: Estimated size of an uncompressed CCNx Content Object

Figure 41 depicts the size requirements for a basic, compressed CCNx Data.

Dispatch Page Switch	= 1	
CCNx Content Dispatch	= 3	
Fixed Header	= 2	
Name		
NameSegments	= n/2 +	
	comps_n	
ExpiryTime	= 8	
Payload	= 1 + clen	
T_HMAC-SHA256	= 32	
SignatureTime	= 8	
ValidationPayload	= 34	
		= 89 + n/2 + comps_n + clen

Figure 41: Estimated size of a compressed CCNx Data Object

The size difference is: $35 + 3.5n$ bytes.

For the name /DE/HH/HAW/BT7, the size is reduced by 70 bytes, which is 40% of the uncompressed packet containing a 4-byte payload.

Acknowledgments

This work was stimulated by fruitful discussions in the ICNRG research group and the communities of RIOT and CCNlite. We would like to thank all active members for constructive thoughts and feedback. In particular, the authors would like to thank (in alphabetical order) Peter Kietzmann, Dirk Kutscher, Martine Lenders, Colin Perkins, Junxiao Shi. The hop-wise stateful name compression was brought up in a discussion by Dave Oran, which is gratefully acknowledged. Larger parts of this work are inspired by [RFC4944] and [RFC6282]. Special mentioning goes to Mark Mosko as well as G.Q. Wang and Ravi Ravindran as their previous work in [TLV-ENC-802.15.4] and [WIRE-FORMAT-CONSID] provided a good base for our discussions on stateless header compression mechanisms. Many thanks also to Carsten Bormann and Lars Eggert, who contributed in-depth comments during the IRSG review. This work was supported in part by the German Federal Ministry of Research and Education within the projects I3 and RAPstore.

Authors' Addresses

Cenk Gundogan
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany

Phone: +4940428758067
Email: cenk.guendogan@haw-hamburg.de
URI: <http://inet.haw-hamburg.de/members/cenk-gundogan>

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany

Email: t.schmidt@haw-hamburg.de
URI: <http://inet.haw-hamburg.de/members/schmidt>

Matthias Waehlich
link-lab & FU Berlin
Hoenower Str. 35
D-10318 Berlin
Germany

Email: mw@link-lab.net
URI: <http://www.inf.fu-berlin.de/~waehl>

Christopher Scherb
University of Basel
Spiegelgasse 1
CH-4051 Basel
Switzerland

Email: christopher.scherb@unibas.ch

Claudio Marxer
University of Basel
Spiegelgasse 1
CH-4051 Basel
Switzerland

Email: claudio.marxer@unibas.ch

Christian Tschudin
University of Basel
Spiegelgasse 1
CH-4051 Basel
Switzerland

Email: christian.tschudin@unibas.ch

ICNRG
Internet-Draft
Intended status: Experimental
Expires: August 3, 2020

G. White
CableLabs
S. Shannigrahi
Tennessee Tech University
C. Fan
Colorado State University
January 31, 2020

Internet Protocol Tunneling over Content Centric Mobile Networks
draft-irtf-icnrg-ipoc-01

Abstract

This document describes a protocol that enables tunneling of Internet Protocol traffic over a Content Centric Network (CCNx) or a Named Data Network (NDN). The target use case for such a protocol is to provide an IP mobility plane for mobile networks that might otherwise use IP-over-IP tunneling, such as the GPRS Tunneling Protocol (GTP) used by the Evolved Packet Core in LTE networks (LTE-EPC). By leveraging the elegant, built-in support for mobility provided by CCNx or NDN, this protocol achieves performance on par with LTE-EPC, equivalent efficiency, and substantially lower implementation and protocol complexity [Shannigrahi]. Furthermore, the use of CCNx/NDN for this purpose paves the way for the deployment of ICN native applications on the mobile network.

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 <https://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 August 3, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	4
3. CCNx Overview	4
4. IPoC Overview	5
4.1. Use of Interest Payloads	5
5. Client Interest Table and Interest Deficit Report	6
6. Handling PIT Entry Lifetimes	7
7. Managing the CIT, PIT lifetimes and the in-flight message count	7
8. Establishing Communication	9
9. IPoC Naming Conventions	9
10. Sequence Numbers	10
11. Packet Sequencer	10
11.1. Packet Sequencer Example Algorithm	11
12. Client Behavior	12
13. Gateway Behavior	12
14. Security Considerations	13
15. IANA Considerations	13
16. References	13
16.1. Normative References	14
16.2. Informative References	14
Authors' Addresses	14

1. Introduction

Content Centric Networking (such as CCNx or NDN, though CCNx is used for the rest of the document) provides some key advantages over IP networking that make it attractive as a replacement for IP for wireless networking. In particular, by employing stateful forwarding, CCNx elegantly supports information retrieval by mobile client devices without the need for tunneling or a location

registration protocol. Furthermore, CCNx supports a client device utilizing multiple network attachments (e.g. multiple radio links) simultaneously in order to provide greater reliability or greater performance. Finally, CCNx is optimized for content retrieval, where content can be easily retrieved from an on-path cache.

From an incremental deployment perspective, it may be attractive to consider supporting CCNx as an overlay, i.e. tunneled over an IP-based mobile core network. But doing so diminishes the value that the CCNx protocol could provide, for example by limiting the ability to utilize on-path caching, native mobility and multiple network attachments. Ultimately, a more powerful approach, one that retains these benefits, is to utilize CCNx as a replacement for IP and IP-over-IP tunneling as the mobility plane for the mobile network.

A significant hurdle that stands in the way of deploying a CCNx-only wireless network is that all of the applications in use today (both client and server) are built to use IP.

This hurdle could be addressed by requiring that all applications be rewritten to use CCNx natively, however, this is a tall order in a world with millions of smartphone apps. Another approach could be to deploy a hybrid network in which the routers support forwarding both IP and CCNx. However, this adds cost and complexity to the network, both in the equipment and in operations.

The protocol described in this document provides a way to eliminate this hurdle, by establishing an IP over CCNx tunneling protocol that is transparent to the IP applications on either end. In a sense, this protocol replaces the IP-over-GTP tunnels or IP-over-GRE tunnels that would exist in a traditional IP-based wireless network such as LTE or Community WiFi, but by using a networking plane (CCNx) that natively supports mobility, application developers have the option to update their applications to run directly over CCNx, gaining all of the advantages that come with this new protocol.

IPoC supports IP mobility within a domain in a manner similar to that supported by LTE-EPC, i.e. the mobile node utilizes an IP address associated with the mobile network to which it is connected, and a stationary gateway device (P-GW in the case of LTE-EPC, IPoC Gateway in the case of IPoC) takes care of forwarding IP packets to the mobile node via the mobile network. [Shannigrahi] compares IPoC to GTP from the perspective of complexity and performance. Other mobility solutions, such as MIPv6 [RFC3775] exist that aim for a broader definition of IP mobility, and support efficient routing even when the mobile device retains an IP address that is not associated with the network to which it is connected. However, all these solutions still inherit the shortcomings of IP networking for

mobility - for example, handover latency and packet loss are known problems with MIPv6 [RFC5268].

This protocol specification does not currently address support for IP multicast connectivity. Support can be achieved via unicast forwarding of IP multicast packets to group members. Other approaches that take CCNx features (such as multicast forwarding strategy and caching) into account could help improve efficiency for IP multicast connectivity.

2. 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].

3. CCNx Overview

In the CCNx protocol, communication is achieved by an application sending an Interest packet that identifies, by name, a piece of content that it wishes to receive. The network routes Interest messages toward a producer of content corresponding to the name in the Interest, leaving a "breadcrumb" trail of state in the routers along that path. Once the Interest arrives at a node where the named piece of content is present, that node returns a Content Object message containing the named piece of content. The Content Object follows (and consumes) the breadcrumb trail back to the originating application. This process is commonly referred to as stateful forwarding. An application that only sends Interest messages is referred to as a consumer, whereas an application that only sends Content Object messages (in response to Interests) is referred to as a producer.

Producers need to advertise the name prefixes for the content that they can provide, and this information needs to propagate to the routers of the network, much in the same way that IP prefixes need to propagate to routers in an IP network. However, consumers don't need to advertise their presence or location at all, they can simply send Interest messages from wherever they are in the network, and the resulting Content Objects will make it back to them via the stateful forwarding process. Furthermore, a consumer that is mobile can redirect data in flight to the its new location by resending Interest messages for those in-flight content objects using its new network attachment point. As a result, mobile consumer applications (which would be the majority of mobile applications) are handled very elegantly by the CCNx protocol.

In addition, if a mobile device has multiple network attachment points, e.g. both a WiFi and a 5G/LTE connection, it can choose to send Interests via both of those network paths. This capability can be used to enable higher capacity (by load balancing the Interests in an attempt to fully utilize multiple links simultaneously), higher reliability (by sending each Interest on multiple links), or seamless handover (by switching to a new link for all future Interest messages, while still waiting to receive Content Objects on an older link).

4. IPoC Overview

While consumer mobility and multipath connectivity is elegantly handled by the CCNx protocol, producer mobility (where a mobile device makes its resident content available to outside devices), is currently not. As a result, the IPoC protocol relies solely on consumer behavior on the client device.

This protocol defines two entities: an IPoC Client and an IPoC Gateway. The IPoC Client (henceforth referred to as the Client) would exist on the mobile device, and as mentioned above, only sends Interest messages. The IPoC Gateway (henceforth referred to as the Gateway) exists at a fixed location in the network, and publishes a prefix that can be routed to via the CCNx network. In general, a network may have many Clients, and possibly several Gateways.

The switches and routers that exist in the path between the Client(s) and the Gateway(s) are assumed to provide CCNx forwarding, and are not required to support IP forwarding.

From the perspective of the IP applications running on the mobile device, the Client implementation functions as a tunnel endpoint, much in the same way that a VPN application does. All IP packets generated by applications on the mobile device are forwarded via this tunnel endpoint, which encapsulates them in CCNx Interest messages, and then sends them into the CCNx network. Similarly, the Gateway implementation also acts as a tunnel endpoint, in this case on an IP routing node. It receives Interest messages, unpacks the IP packets inside, and forwards them into an IP network. IP return traffic arriving at the Gateway is encapsulated into CCNx Content Object messages, and then launched into the CCNx network to follow the stateful forwarding path left by the associated Interest message.

4.1. Use of Interest Payloads

As described above, IPoC capitalizes on the consumer mobility features of CCNx, and as a result uses the optional interest payload mechanism described in the "Consumer Behavior" section of [RFC8569].

This behavior preserves the basic hop-by-hop flow balancing principle of ICN, in that intermediate routers can control traffic flow by delaying Interest messages as appropriate. Additionally, the interest payload allows transport of information in Interests outside of the name field, which can significantly reduce router complexity (memory and memory bandwidth), as the name field is stored in the router's Pending Interest Table.

5. Client Interest Table and Interest Deficit Report

In this communication model, the Client is able to send "upstream" packets at any time, by sending Interest messages. The Gateway on the other hand, can only send "downstream" packets when it has a pending Interest (i.e. it has received an Interest message and has not yet responded with an associated Content Object). As a result, the Client and Gateway work together to ensure that the Gateway is receiving Interests sufficiently to support the downstream communication.

For each Client, the Gateway MUST maintain a FIFO queue of names for which it has received Interests from the Client. This queue is referred to as the Client Interest Table (CIT). As this is a FIFO queue, the order in which Interest names are received is the order in which the associated Content Object responses will be sent.

The typical behavior of a Client (described in more detail below) is to send an Interest message for every Content Object it receives, thus maintaining a constant number of CCNx packets "in flight". The Interest Deficit Report (IDR) is a message element sent in a Content Object from the Gateway to the Client in order to adjust the number of packets in flight and thus maintain an appropriate CIT size. The IDR can take the value +1, to request an increase (by one) of the in-flight count; 0 to indicate no change to the in-flight count; or -1 to request a decrease (by one) of the in-flight count. The IDR can be included in a Content Object that carries a packet payload, or in a Content Object that is otherwise empty.

The IDR is an unacknowledged message element, and as such is an inherently unreliable communication. Since the IDR values are small, the impact of a Content Object loss is minimal.

The Client MUST maintain an Interest Deficit Count (IDC) which it uses to maintain the in-flight count in response to sent Interests and received Content Objects. The Client MUST decrement by one the IDC upon transmission of a new Interest message. The Client MUST update the IDC by adding IDR+1 to its value upon receipt of a new Content Object.

The Gateway SHOULD NOT discard Interest names from the CIT, and thus SHOULD always respond to a received Interest with a Content Object in order to clear the associated PIT state in the intermediate routers. If a new Interest arrives and the CIT is full, the gateway MUST consume the name at the head of the CIT by sending an empty content object. In this case, the IDR value of the empty Content Object SHOULD be set to -1.

6. Handling PIT Entry Lifetimes

Intermediate routers between the Client and the gateway, as well as CCNx forwarder implementations within the two IPoC endpoints will store PIT entries for the Client's Interests for a finite lifetime, and will age-out (purge) Interests that exceed that lifetime. Since the CIT at the gateway stores Interest names for a time in anticipation of downstream packets, it would be possible, when there is a gap in the flow of downstream packets, that the name at the head of the CIT queue is associated with entries that have been aged-out of the PIT in one or more of the intermediate forwarders. If the gateway were to use this aged-out name in an attempt to deliver a downstream packet, the packet transmission would fail when the Content Object arrived at the PIT that no longer held an entry for this name.

To avoid this situation, the Gateway MUST record the arrival time of each CIT entry, and compare it against a CIT lifetime value. When the CIT entry at the head of the CIT "expires", the gateway MUST send a Content Object using that CIT entry, thereby cleaning up the PIT state in the intervening forwarders, and potentially triggering a new Interest to be sent by the Client (as discussed further below).

7. Managing the CIT, PIT lifetimes and the in-flight message count

At any instant in time, a certain number of Interest names can be considered "in-flight" from the Client's perspective (these in-flight Interests correspond to the entries in the Client's PIT). Some fraction of the in-flight Interest names will correspond to Interest messages (possibly containing IP packets) that are in transit to the gateway, some fraction will correspond to Content Object messages (also possibly containing IP packets) that are in transit to the Client, and the remainder correspond to the entries in the gateway's CIT or to messages that were lost in transit. The gateway controls the number of these in-flight messages via the IDR, which can either trigger or suppress the Client sending Interests.

Since the gateway cannot send a downstream packet to the Client unless it has a CIT entry, it would ideally like to ensure that it always has at least one CIT entry every time a downstream packet

arrives. However, due to the round trip time between the gateway and the Client, and the fluctuation of downstream and upstream packet arrival rates, the number of in-transit messages (Interests or Content Objects) will fluctuate. If the only goal was that the CIT never becomes empty, the gateway could simply use the IDR to build a very high in-flight message count. This would ensure that the CIT never drains completely, even in the case where the upstream path and the downstream path are both saturated with in-transit messages. The problem with this approach is that when the connection becomes idle, ALL of the in-flight messages would then exist in the CIT, which could be a large memory burden on the gateway and on the PIT in each intervening router. Furthermore, since each of these CIT entries has a certain lifetime, driven by the PIT lifetime, they will shortly expire, triggering the gateway to transmit Content Objects that heavily utilize the downstream and upstream links for approximately one RTT. This pattern of unnecessary network traffic would then periodically repeat at a period equal to the CIT lifetime.

So, it is important that the gateway adjust the in-flight message count continuously, to minimize the times that the CIT is starved or flooded.

The gateway MUST establish a target minimum value for the number of CIT entries. This value "n" provides a bound on the number of downstream packets that can be sent in the first IPoC RTT (between gateway and client) after an idle period, and also establishes the quiescent IPoC message refresh rate during idle periods (this rate $r = n/L$, where L is the CIT lifetime). Selecting a low value of n minimizes the quiescent load on the network, but has the downside of reducing the size of packet burst that the IPoC connection can handle with low latency.

Whenever the gateway sends a Content Object and there are fewer than n CIT entries, it MUST include an IDR in the CO, with the value 1, triggering the Client to send two Interest messages in response to the CO.

The gateway also MUST establish a maximum CIT size "N". Whenever the gateway receives a new Interest while the CIT contains N entries, it MUST make room for the new CIT entry by using the head of line CIT entry to send an empty Content Object containing an IDR with the value -1, triggering the Client to suppress sending an Interest in response.

Further, whenever the CIT entry at the head of line expires (reaches its CIT lifetime), the Gateway MUST consume that CIT entry by sending an empty Content Object. The expiration of a CIT entry is a good indication that the CIT contains more entries than are needed to

support the current data rate. In this situation, the Gateway SHOULD use the IDR to reduce the in-flight count. One mechanism for doing this is described here:

If the number of CIT entries is less than n , the empty Content Object sent to consume the expiring CIT entry will contain an IDR with the value 1. If the number of CIT entries is greater than n , the CO will contain an IDR with value -1, and if it is equal to n , the value 0. The result of this process is that during idle periods, the CIT will drain down to the point of having n entries, and will refresh those entries as they expire.

8. Establishing Communication

Communication is established by the Client sending an Interest to a Gateway, where the name in the Interest message includes a Gateway prefix followed by `/init/<random_string>`. For example, if the established Gateway prefix is `ccnx:/ipoc`, the name might be `ccnx:/ipoc/init/2Fhwte2452g5shH4`. The Gateway has a process that will respond to the `ccnx:/ipoc/init` prefix by sending IP configuration information, similar to the information contained in a DHCP Offer, including an assigned IP address.

Upon configuring itself using the information in the init response, the Client can begin IP communication. The naming convention for subsequent Interest messages is described in the next section.

9. IPoC Naming Conventions

The IPoC protocol doesn't assign any relationship between the Interest / Content Object names and the contents of the encapsulated IP packets. Rather, the name only identifies the Client instance of the IPoC application, and provides a sequence number that disambiguates Interests and Content Objects and provides for in-order delivery of IP packets.

The Client and Gateway can use one of the following data naming conventions, the appropriate naming convention is chosen by the Gateway via configuration, and is communicated to the Client during the Establishing Communication protocol.

`ccnx:/ipoc/<hex_ipaddr>/<b64_seq>`

`ccnx:/ipoc/<zone_id>/<hex_ipaddr>/<b64_seq>`

The various components of an IPoC name are described in more detail below:

- o ccnx:/ipoc - The name prefix used in all IPoC messages
- o zone_id - An optional zone identifier to allow for zone-based IP address re-use.
- o hex_ipaddr - For IPv4 addresses, this field comprises 4 separate name segments, each representing a single octet of an IPv4 address encoded as a hexadecimal string. For example, a message from a Client with IPv4 address 192.0.2.100 would use: "c0/00/02/64" for this name component. For IPv6 addresses, the textual convention defined in Section 2.2 paragraph 1 of [RFC4291] is used, with each colon replaced by a CCNx name segment delimiter. For example a Client with the IPv6 address: 2001:DB8::fe21:67cf would use "2001/DB8/0/0/0/0/fe21/67cf" for this name component.
- o b64_seq - This a base64-encoded value representing the Upstream Sequence Number for this upstream Interest message

An example Interest name is: ccnx:/ipoc/c0/00/02/64/AAAAGw==

10. Sequence Numbers

Upstream Sequence Numbers (USN) are monotonically increasing unsigned 32-bit integer values embedded in the Interest names to indicate the proper ordering for upstream data packets. Since Interest messages may arrive out-of-order due to the use of multiple network paths, the Gateway uses the USN to ensure that upstream IP packets are delivered in the proper order.

Content Objects that carry IP packet payloads include Downstream Sequence Numbers (DSN), which are monotonically increasing unsigned 32-bit integer values that indicate the proper ordering of downstream data packets. DSN are used by the Client to ensure that downstream IP packets are delivered in the proper order.

The USN and DSN are independent sequence numbers and thus have no relationship to one another.

11. Packet Sequencer

The Packet Sequencer (PS or Sequencer) is a FIFO queue that exists both at the Client and Gateway to ensure in-order delivery of IP packets contained in upstream Interests and downstream Content Objects. The order in which the packets are delivered is decided by the Packet Sequence Number (PSN) embedded in the Interest or Content Object names.

The client MUST implement a Packet Sequencer to ensure in-order delivery of IP packets. The gateway MUST implement a Packet Sequencer to ensure in-order delivery of IP packets.

11.1. Packet Sequencer Example Algorithm

The first PSN (FPSN) delivered to the Sequencer establishes a baseline to which all subsequent PSNs are evaluated based on an expected ascending incremental order. The Sequencer also notes the last PSN (LPSN) it forwarded, and for the first packet, FPSN is equal to LPSN. If an arriving packet has the expected sequence number (LPSN + 1), the sequencer does not queue the packet and simply forwards it. The Sequencer also tracks the highest sequence number that has arrived (MAXPSN).

Discontinuities in the sequence order result in a "gap" in the sequence. If the arriving packet has a sequence number LPSN + n, where $n > 1$, we declare this as a gap. For example, if the last forwarded PSN had a sequence number 6 (LPSN), and a new packet arrives with sequence number 10 (MAXPSN), a new gap is created which represents the sequence numbers 7, 8, and 9. A timer with a validity window is started providing a limited amount of time for the sequence numbers in the gap to arrive.

Each time a packet with a sequence number in the gap arrives, the Sequencer tries to do a partial release of the queue; this releases any consecutive packets between LPSN and MAXPSN. In our example, if sequence 8 arrives first, the Sequencer sees there are no consecutive packets to send and does nothing. If sequence 7 arrives after that, the Sequencer releases both 7 and 8 but waits for sequence 9. When sequence 9 arrives, it releases 9 and 10. If a packet does not arrive and the validity window expires, the Sequencer releases all packets up to MAXPSN and reset the LPSN.

The sequencer removes data packets from the queue in sequence-order (lowest PSN first). If the queue exceeds capacity, the Sequencer discards the packet with the lowest PSN. Any IP packets in those Interests or content objects are discarded.

Ideally, the gap validity window should be set to the RTT between the Client and the Gateway. However, since packets can take multiple paths and the Sequencer may not know the RTT for each of these paths, it should dynamically adjust the validity window based on the inter-arrival time between consecutive packets.

12. Client Behavior

The three main functions of the Client are:

1. Send Interest messages containing upstream IP packets whenever they arrive
2. Send Interest messages to the gateway in order to keep the appropriate in-flight count
3. Receive downstream IP packet data in Content Object messages

Content Object messages containing downstream IP packet data are added to the Packet Sequencer and then forwarded to the IP stack on the device.

Once an IP address is acquired using the initialization process described above, the startup sequence for a particular Client looks like this:

- o Initialize IDC to a startup value: INIT_IDC.
- o Send Interest messages to the Gateway containing the initial upstream IP packets (e.g. TCP SYN packets or DNS queries), decrementing IDC for each Interest sent.

The client MUST decrement the IDC upon transmission of any Interest message, whether or not it contains an upstream packet.

Whenever the client receives a Content Object, it MUST increment the IDC by IDR+1 to ensure that the appropriate in-flight count is maintained.

The Client MUST maintain two internal timer intervals. A short timer (T0) is used to pace Interest messages when there are outstanding interests to be sent as per the Interest Deficit Counter. The long timer (T1) is used as a keep-alive when the Client has no outstanding Interests to be sent. Whenever the client sends an Interest message, it restarts the T0 and T1 timers. When the T0 timer expires, if the IDC is greater than zero, the Client MUST send an empty Interest message. When the T1 timer expires, the Client MUST send an empty Interest message (regardless of the IDC value).

13. Gateway Behavior

IPoC gateway behavior is slightly more complex since it must manage connections with multiple Clients simultaneously. The standard process for on-boarding a new Client looks something like this:

- o An Interest is received with the /init/<random_string> name.
- o The gateway establishes new CIT (and other Client-specific) structures for this Client and responds with a Content Object containing the IP parameters (yiaddr, giaddr, etc.) to configure the Client's IP stack.
- o The gateway enters a normal processing loop in which it receives Interests from the Client and responds with Content Objects.

Interests received from the Client may contain IP packets that the gateway will add to its upstream Packet Sequencer using the PSN found in the Interest name. The Interest name will then be added the Client-specific CIT for later use in creating Content Objects. If the CIT is full, the gateway will immediately send an empty Content Object back to the Client, removing the first name from the CIT, and therefore making room for the new name to be added.

When downstream IP packets become available, the gateway will remove the first name from the CIT queue and use it to create a Content Object containing the IP packets. If the CIT is empty, IP packets are buffered by the gateway.

If IP packets are waiting in buffer when a new Interest (CIT entry) arrives, the gateway will immediately dequeue the waiting packets (up to a maximum CO size limit), form and transmit a Content Object using the newly arrived CIT name.

14. Security Considerations

This protocol is designed for use within a trusted domain (i.e. a mobile core network). This protocol definition does not address authentication between clients and gateway devices, nor does it address privacy of communications (beyond that already provided by the IP applications themselves). The CCNx protocol does provide for Interest message and Content Object message authentication (signing) [RFC8609], which can be utilized if desired.

15. IANA Considerations

This document has no actions for IANA.

16. References

16.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, DOI 10.17487/RFC3775, June 2004, <<https://www.rfc-editor.org/info/rfc3775>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC5268] Koodli, R., Ed., "Mobile IPv6 Fast Handovers", RFC 5268, DOI 10.17487/RFC5268, June 2008, <<https://www.rfc-editor.org/info/rfc5268>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.

16.2. Informative References

- [Shannigrahi] Shannigrahi, S., Fan, C., and G. White, "Bridging the ICN Deployment Gap with IPoC: An IP-over-ICN protocol for 5G Networks", SIGCOMM NEAT Workshop, August 2018, <<https://dl.acm.org/citation.cfm?id=3229575>>.

Authors' Addresses

Greg White
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
US

Email: g.white@cablelabs.com

Susmit Shannigrahi
Tennessee Tech University
Computer Sc. Dept.
Cookeville, TN 38501
US

Email: sshannigrahi@tntech.edu

Chengyu Fan
Colorado State University
Computer Sc. Dept.
1100 Center Ave Mall
Ft. Collins, CO 80523
US

Email: chengyu.fan@colostate.edu

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: 29 January 2022

J. Hong
T. You
ETRI
L. Dong
C. Westphal
Futurewei Technologies Inc.
B. Ohlman
Ericsson
28 July 2021

Design Considerations for Name Resolution Service in ICN
draft-irtf-icnrg-nrs-requirements-06

Abstract

This document provides the functionalities and design considerations for a Name Resolution Service (NRS) in ICN. An NRS in ICN is to translate an object name into some other information such as a locator, another name, etc. for forwarding the object request. This document is a product of the Information-Centric Networking Research Group (ICNRG).

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 <https://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 29 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Name Resolution Service in ICN	4
2.1. Explicit name resolution approach	4
2.2. Name-based routing approach	4
2.3. Hybrid approach	5
2.4. Comparisons of name resolution approaches	5
3. Functionalities of NRS in ICN	6
3.1. Support heterogeneous name types	7
3.2. Support producer mobility	8
3.3. Support scalable routing system	9
3.4. Support off-path caching	10
3.5. Support nameless object	10
3.6. Support manifest	11
3.7. Support metadata	11
4. Design considerations for NRS in ICN	11
4.1. Resolution response time	11
4.2. Response accuracy	12
4.3. Resolution guarantee	12
4.4. Resolution fairness	12
4.5. Scalability	13
4.6. Manageability	14
4.7. Deployed system	14
4.8. Fault tolerance	14
4.9. Security and privacy	14
4.9.1. Confidentiality	14
4.9.2. Authentication	15
4.9.3. Integrity	15
4.9.4. Resiliency and availability	15
5. Conclusion	16
6. IANA Considerations	16
7. Security Considerations	16
8. References	16
8.1. Normative References	16
8.2. Informative References	16
Acknowledgements	19
Authors' Addresses	19

1. Introduction

The current Internet is based upon a host-centric networking paradigm, where hosts are identified with IP addresses and communication is possible between any pair of hosts. Thus, information in the current Internet is identified by the name of the host (or server) where information is stored. In contrast to host-centric networking, the primary communication objects in Information-centric networking (ICN) are the named data objects (NDOs) and they are uniquely identified by location-independent names. Thus, ICN aims for the efficient dissemination and retrieval of NDOs at a global scale, and has been identified and acknowledged as a promising technology for a future Internet architecture to overcome the limitations of the current Internet such as scalability and mobility [Ahlgren] [Xylomenos]. ICN also has emerged as a candidate architecture in the IoT environment since IoT focuses on data and information [Baccelli] [Amadeo] [Quevedo] [Amadeo2] [ID.Zhang2].

Since naming data independently from its current location (where it is stored) is a primary concept of ICN, how to find any NDO using a location-independent name is one of the most important design challenges in ICN. Such ICN routing may comprise three steps [RFC7927]:

- * Name resolution: matches/translate a content name to the locator of content producer or source that can provide the content.
- * Content request routing: routes the content request towards the content's location either based on its name or locator.
- * Content delivery: transfers the content to the requester.

Among the three steps of ICN routing, this document investigates only the name resolution step which translates a content name to the content locator. In addition, this document covers various possible types of name resolution in ICN such as one name to another name, name to locator, name to manifest, name to metadata, etc.

The focus of this document is a Name Resolution Service (NRS) itself as a service or a system in ICN and it provides the functionalities and the design considerations for an NRS in ICN as well as the overview of the NRS approaches in ICN. On the other hand, its companion document [NRSarch] describes considerations from the perspective of ICN architecture and routing system when using an NRS in ICN.

This document represents the consensus of the Information-Centric Networking Research Group (ICNRG). It has been reviewed extensively by the Research Group (RG) members who are actively involved in the research and development of the technology covered by this document. It is not an IETF product and is not a standard.

2. Name Resolution Service in ICN

A Name Resolution Service (NRS) in ICN is defined as the service that provides the name resolution function for translating an object name into some other information such as a locator, another name, metadata, next hop info, etc. that is used for forwarding the object request. In other words, an NRS is a service that can be provided by the ICN infrastructure to help a consumer to reach a specific piece of information (or named data object). The consumer provides an NRS with a persistent name and the NRS returns a name or locator (or potentially multiple names and locators) that can reach a current instance of the requested object.

The name resolution is a necessary process in ICN routing although the name resolution either can be separated from the content request routing as an explicit process or can be integrated with the content request routing as an implicit process. The former is referred as explicit name resolution approach, the latter is referred as name-based routing approach in this document.

2.1. Explicit name resolution approach

An NRS could take the explicit name resolution approach to return the locators of the content to the client, which will be used by the underlying network as the identifier to route the client's request to one of the producers or to a copy of the content. There are several ICN projects that use the explicit name resolution approach such as DONA [Koponen], PURSUIT [PURSUIT], NetInf [SAIL], MobilityFirst [MF], IDNet [Jung], etc. In addition, the explicit name resolution approach has been allowed for 5G control planes [SA2-5GLAN].

2.2. Name-based routing approach

An NRS could take the name-based routing approach, which integrates the name resolution with the content request message routing as in NDN [NDN]/CCNx [CCNx].

In the case that the content request also specifies the reverse path, as in NDN/CCNx, the name resolution mechanism also derives the routing path for the data. This adds a requirement on the name resolution service to propagate request in a way that is consistent with the subsequent data forwarding. Namely, the request must select a path for the data based upon finding a copy of the content, but also properly delivering the data.

2.3. Hybrid approach

An NRS could also take hybrid approach. For instance, it can attempt the name-based routing approach first. If this fails at a certain router, the router can go back to the explicit name resolution approach. The hybrid NRS approach also works the other way around by performing explicit name resolution first to find locators of routers. And then it can carry out the name-based routing approach of the client's request.

A hybrid approach would combine name resolution over a subset of routers on the path with some tunneling in between (say, across an administrative domain) so that only a few of the nodes in the ICN network perform name resolution in the name-based routing approach.

2.4. Comparisons of name resolution approaches

The following compares the explicit name resolution and the name-based routing approaches for several aspects:

- * Overhead due to the maintenance of the content location: The content reachability is dynamic and includes new content being cached or content being expired from a cache, content producer mobility, etc. Maintaining a consistent view of the content location across the network requires some overhead that differs for the name resolution approaches. The name-based routing approach may require flooding parts of the network for update propagation. In the worst case, the name-based routing approach may flood the whole network (but mitigating techniques may be used to scope the flooding). However, the explicit name resolution approach only requires updating propagation in part of the name resolution system (which could be an overlay with a limited number of nodes).
- * Resolution capability: The explicit name resolution approach, if designed and deployed with sufficient robustness, can offer at least weak guarantees that resolution will succeed for any content name in the network if it is registered to the name resolution overlay. In the name-based routing approach, content resolution depends on the flooding scope of the content names (i.e. content

publishing message and the resulting name-based routing tables). For example, when a content is cached, the router may only notify this information to its direct neighbors. Thus, only those neighboring routers can build a named based entry for this cached content. But if the neighboring routers continue to propagate this information, the other nodes are able to direct to this cached copy as well.

- * **Node failure impact:** Nodes involved in the explicit name resolution approach are the name resolution overlay servers (e.g. Resolution Handlers in DONA), while the nodes involved in the name-based routing approach are routers which route messages based on the name-based routing tables (e.g. NDN routers). Node failures in the explicit name resolution approach may cause some content request routing to fail even though the content is available. This problem does not exist in the name-based routing approach because other alternative paths can be discovered to bypass the failed ICN routers, given the assumption that the network is still connected.
- * **Maintained databases:** The storage usage for the explicit name resolution approach is different from that of the name-based routing approach. The explicit name resolution approach typically needs to maintain two databases: name to locator mapping in the name resolution overlay and routing tables in the routers on the data forwarding plane. The name-based routing approach needs to maintain only the name-based routing tables.

Additionally, some other intermediary step may be included in the name resolution, namely the mapping of one name to other names, in order to facilitate the retrieval of named content, by way of a manifest [Westphal] [Mosko]. The manifest is resolved using one of the two above approaches, and it may include further mapping of names to content and location. The steps for name resolution then become: first translate the manifest name into a location of a copy of the manifest; the manifest includes further names of the content components, and potentially locations for the content. The content is then retrieved by using these names and/or location, potentially resulting in additional name resolutions.

Thus, no matter which approach is taken by an NRS in ICN, the name resolution is the essential function that shall be provided by the ICN infrastructure.

3. Functionalities of NRS in ICN

This section presents the functionalities of an NRS in ICN.

3.1. Support heterogeneous name types

In ICN, a name is used to identify data object and is bound to it [RFC7927]. ICN requires uniqueness and persistency of the name of data object to ensure the reachability of the object within a certain scope. There are heterogeneous approaches to designing ICN naming schemes [Bari]. Ideally, a name can include any form of identifier, which can be flat or hierarchical, and human readable or non-readable.

Although there are diverse types of naming schemes proposed in literature, they all need to provide basic functions for identifying data object, supporting named data lookup and routing. An NRS may combine the better aspects of different schemes. Basically, an NRS should be able to support a generic naming schema so that it can resolve any type of content name, irrespective of whether it is flat, hierarchical, attribute-based or anything else.

In PURSUIT [PURSUIT], names are flat and the rendezvous functions are defined for an NRS, which is implemented by a set of Rendezvous Nodes (RNs), the Rendezvous Network (RENE). Thus, a name consists of a sequence of scope IDs and a single rendezvous ID is routed by the RNs in RENE. Thus, PURSUIT decouples name resolution and data routing, where the NRS is performed by the RENE.

In MobilityFirst [MF], a name called a Global Unique Identifier (GUID) derived from a human-readable name via a global naming service is a flat typed 160-bits string with self-certifying properties. Thus, MobilityFirst defines a Global Name Resolution Service (GNRS) which resolves GUIDs to network addresses and decouples name resolution and data routing similarly to PURSUIT.

In NetInf [Dannewitz], information objects are named using ni-naming [RFC6920], which consist of an authority part and digest part (content hash). The ni names can be flat as the authority part is optional. Thus, the NetInf architecture also includes a Name Resolution System (NRS) which can be used to resolve ni-names to addresses in an underlying routable network layer.

In NDN [NDN] and CCNx [CCNx], names are hierarchical and may be similar to URLs. Each name component can be anything, including a human-readable string or a hash value. NDN/CCNx adopts the name-based routing approach. The NDN router forwards the request by doing the longest-match lookup in the Forwarding Information Base (FIB) based on the content name and the request is stored in the Pending Interest Table (PIT).

3.2. Support producer mobility

ICN natively supports mobility management. Namely, consumer or client mobility is handled by re-requesting the content in case the mobility event (say, handover) occurred before receiving the corresponding content from the network. Since ICN can ensure that content reception continues without any disruption in ICN applications, seamless mobility from the consumer's point of view can be easily supported.

However, producer mobility does not emerge naturally from the ICN forwarding model as does consumer mobility. If a producer moves into a different network location or a different name domain, which is assigned by another authoritative publisher, it would be difficult for the mobility management to update RIB and FIB entries in ICN routers with the new forwarding path in a very short time. Therefore, various ICN architectures in the literature have proposed to adopt an NRS to achieve the producer or publisher mobility, where the NRS can be implemented in different ways such as rendezvous points and/or overlay mapping systems.

In NDN [Zhang2], for producer mobility support, rendezvous mechanisms have been proposed to build interests rendezvous (RV) with data generated by a mobile producer (MP). This can be classified into two approaches: chase mobile producer; and rendezvous data. Regarding MP chasing, rendezvous acts as a mapping service that provides the mapping from the name of the data produced by the MP to the name of the MP's current point of attachment (PoA). Alternatively, the RV serves as a home agent as in IP mobility support, so the RV enables consumer's interest message to tunnel towards the MP at the PoA. Regarding rendezvous data, the solution involves moving the data produced by the MP to a data depot instead of forwarding interest messages. Thus, a consumer's interest message can be forwarded to stationary place as called data rendezvous, so it would either return the data or fetch it using another mapping solution. Therefore, RV or other mapping functions are in the role of an NRS in NDN.

In [Ravindran], forwarding-label (FL) object is referred to enable identifier (ID) and locator (LID) namespaces to be split in ICN. Generally, IDs are managed by applications, while locators are managed by a network administrator, so that IDs are mapping to heterogeneous name schemes and LIDs are mapping to the network domains or to specific network elements. Thus, the proposed FL object acts as a locator (LID) and provides the flexibility to forward Interest messages through mapping service between IDs and LIDs. Therefore, the mapping service in control plane infrastructure can be considered as an NRS in this draft.

In MobilityFirst [MF], both consumer and publisher mobility can be primarily handled by the global name resolution service (GNRS) which resolves GUIDs to network addresses. Thus, the GNRS must be updated for mobility support when a network attached object changes its point of attachment, which differs from NDN/CCNx.

In NetInf [Dannewitz], mobility is handled by an NRS in a very similar way to MobilityFirst.

Besides the consumer and producer mobility, ICN also has to face challenges to support the other dynamic features such as multi-homing, migration, and replication of named resources such as content, devices, and services. Therefore, an NRS can help to support these dynamic features.

3.3. Support scalable routing system

In ICN, the name of data objects is used for routing by either a name resolution step or a routing table lookup. Thus, routing information for each data object should be maintained in the routing base, such as Routing Information Base (RIB) and Forwarding Information Base (FIB). Since the number of data objects would be very large, the size of information bases would be significantly larger as well [RFC7927].

The hierarchical namespace used in CCNx [CCNx] and NDN [NDN] architectures reduces the size of these tables through name aggregation and improves the scalability of the routing system. A flat naming scheme, on the other hand, would aggravate the scalability problem of the routing system. The non-aggregated name prefixes injected to the Default Route Free Zone (DFZ) of ICN would create more serious scalability problem when compared to the scalability issues of the IP routing system. Thus, an NRS may play an important role in the reduction of the routing scalability problem regardless of the types of namespaces.

In [Afanasyev], in order to address the routing scalability problem in NDN's DFZ, a well-known concept of Map-and-Encap is applied to provide a simple and secure namespace mapping solution. In the proposed map-and-encap design, data whose name prefixes do not exist in the DFZ forwarding table can be retrieved by a distributed mapping system called NDNS, which maintains and lookups the mapping information from a name to its globally routed prefixes, where NDNS is a kind of an NRS.

3.4. Support off-path caching

Caching in-network is considered to be a basic architectural component of an ICN architecture. It may be used to provide a level of Quality-of-Service (QoS) experience to users, to reduce the overall network traffic, to prevent network congestion and Denial-of-Service (DoS) attacks and to increase availability. Caching approaches can be categorized into off-path caching and on-path caching based on the location of caches in relation to the forwarding path from the original server to the consumer. Off-path caching, also referred as content replication or content storing, aims to replicate content within a network in order to increase availability, regardless of the relationship of the location to the forwarding path. Thus, finding off-path cached objects is not trivial in name-based routing of ICN. In order to support off-path caches, replicas are usually advertised into a name-based routing system or into an NRS.

In [Bayhan], an NRS is used to find off-path copies in the network, which may not be accessible via name-based routing mechanisms. Such capability can be helpful for an Autonomous System (AS) to avoid the costly inter-AS traffic for external content more, to yield higher bandwidth efficiency for intra-AS traffic, and to decrease the data access latency for a pleasant user experience.

3.5. Support nameless object

In CCNx 1.0 [Mosko2], the concept of "Nameless Objects" that are a Content Object without a Name is introduced to provide a means to move Content between storage replicas without having to rename or resign the content objects for the new name. Nameless Objects can be addressed by the ContentObjectHash that is to restrict Content Object matching by using SHA-256 hash.

An Interest message would still carry a Name and a ContentObjectHash, where a Name is used for routing, while a ContentObjectHash is used for matching. However, on the reverse path, if the Content Object's name is missing, it is a "Nameless Object" and only matches against the ContentObjectHash. Therefore, a consumer needs to resolve proper name and hashes through an outside system, which can be considered as an NRS.

3.6. Support manifest

For collections of data objects which are organized as large and file like contents [FLIC], manifests are used as data structures to transport this information. Thus, manifests may contain hash digests of signed content objects or other manifests, so that large content objects which represent large piece of application data can be collected by using such manifest.

In order to request content objects, a consumer needs to know a manifest root name to acquire the manifest. In case of FLIC, a manifest name can be represented by a nameless root manifest, so that outside system such as an NRS may be involved to give this information to the consumer.

3.7. Support metadata

When resolving the name of a content object, NRS could return a rich set of metadata in addition to returning a locator. The metadata could include alternative object locations, supported object transfer protocol(s), caching policy, security parameters, data format, hash of object data, etc. The consumer could use this metadata for selection of object transfer protocol, security mechanism, egress interface, etc. An example of how metadata can be used in this way is provided by the NEO ICN architecture [NEO].

4. Design considerations for NRS in ICN

This section presents the design considerations for NRS in ICN.

4.1. Resolution response time

The name resolution process should provide a response within a reasonable amount of time. The response should be either a proper mapping of the name to a copy of the content, or an error message stating that no such object exists. If the name resolution does not map to a location, the system may not issue any response, and the client should set a timer when sending a request, so as to consider the resolution incomplete when the timer expires.

The acceptable response delay could be of the order of a round trip time between the client issuing the request and the NRS servers that provides the response. While this RTT may vary greatly depending on the proximity between the two end points, some upper bound need be used. Especially, in some delay-sensitive scenarios such as industrial Internet and telemedicine, the upper bound of the response delay must be guaranteed.

The response time includes all the steps of the resolution, including potentially a hop-by-hop resolution or a hierarchical forwarding of the resolution request.

4.2. Response accuracy

An NRS must provide an accurate response, namely a proper binding of the requested name (or prefix) with a location. The response can be either a (prefix, location) pair, or the actual forwarding of a request to a node holding the content, which is then transmitted in return.

An NRS must provide an up-to-date response, namely an NRS should be updated within a reasonable time when new copies of the content are being stored in the network. While every transient cache addition/eviction should not trigger an NRS update, some origin servers may move and require the NRS to be updated.

An NRS must provide mechanisms to update the mapping of the content with its location. Namely, an NRS must provide a mechanism for a content provider to add new content, revoke old/dated/obsolete content, and modify existing content. Any content update should then be propagated through the NRS system within reasonable delay.

Content that is highly mobile may require to specify some type of anchor that is kept at the NRS, instead of the content location.

4.3. Resolution guarantee

An NRS must ensure that the name resolution is successful with high probability if the name matching content exists in the network, regardless of its popularity and number of cached copies existing in the network. As per Section 4.1, some resolution may not occur in a timely manner. However, the probability of such event should be minimized. The NRS system may provide a probability (say, five 9s, or five sigmas for instance) that a resolution will be satisfied.

4.4. Resolution fairness

An NRS could provide this service for all content in a fair manner, independently of the specific content properties (content producer, content popularity, availability of copies, content format, etc.). Fairness may be defined as a per request delay to complete the NRS steps that is not agnostic to the properties of the content itself. Fairness may be defined as well as the number of requests answered per unit of time.

However, it is notable that content (or their associated producer) may request a different level of QoS from the network (see [RFC9064] for instance), and this may include the NRS as well, in which case considerations of fairness may be restricted to content within the same class of service.

4.5. Scalability

The NRS system must scale up to support a very large user population (including human users as well as machine-to-machine communications). As an idea of the scale, it is expected that 50 billion devices will be connected in 2025 (per ITU projections). The system must be able to respond to a very large number of requests per unit of time. Message forwarding and processing, routing table building-up and name records propagation must be efficient and scalable.

The NRS system must scale up with the number of pieces of content (content names) and should be able to support a content catalog that is extremely large. Internet traffic is of the order of the zettabytes per year (10^{21} bytes). Since NRS is associated with actual traffic, the number of pieces of content should scale with the amount of traffic. Content size may vary from a few bytes to several GB, so the NRS should be expected scale up to catalog of the size of 10^{21} in the near future, and larger beyond.

The NRS system must be able to scale up, namely to add NRS servers to the NRS system, in a way that is transparent to the users. Addition of a new server should have limited negative impact on the other NRS servers (or should have a negative impact on only a small subset of the NRS servers). The impact of adding new servers may induce some overhead at the other servers to rebuild a hierarchy or to exchange messages to include the new server within the service. Further, data may be shared among the new servers, for load balancing or tolerance to failure. These steps should not disrupt the service provided by the NRS and should in the long run improve the quality of the service.

The NRS system may support access from a heterogeneity of connection methods and devices. In particular, the NRS system may support access from constrained devices and interactions with the NRS system would not be too costly. An IoT node for instance should be able to access the NRS system as well as a more powerful node.

The NRS system should scale up in its responsiveness to the increased request rate that is expected from applications such as IoT or M2M, where data is being frequently generated and/or frequently requested.

4.6. Manageability

The NRS system must be manageable since some parts of the system may grow or shrink dynamically and an NRS system node may be added or deleted frequently.

The NRS system may support an NRS management layer that allows for adding or subtracting NRS nodes. In order to infer the circumstance, the management layer can measure network status.

4.7. Deployed system

The NRS system must be deployable since deployability is important for a real-world system. The NRS system must be deployable in network edges and cores so that the consumers as well as ICN routers can perform name resolution in a very low latency.

4.8. Fault tolerance

The NRS system must ensure resiliency in the event of NRS server failures. The failure of a small subset of nodes should not impact the NRS performance significantly.

After an NRS server fails, the NRS system must be able to recover and/or restore the name records stored in the NRS server.

4.9. Security and privacy

On utilizing an NRS in ICN, there are some security considerations for the NRS servers/nodes and name mapping records stored in the NRS system. This subsection describes them.

4.9.1. Confidentiality

The name mapping records in the NRS system must be assigned with proper access rights such that the information contained in the name mapping records would not be revealed to unauthorized users.

The NRS system may support access control for certain name mapping records. Access control can be implemented with a reference monitor that uses client authentication, so only users with appropriate credentials can access these records, and they are not shared with unauthorized users. Access control can also be implemented by encryption-based techniques using control of keys to control the propagations of the mappings.

The NRS system may support obfuscation and/or encryption mechanisms so that the content of a resolution request may not be accessible by third parties outside of the NRS system.

The NRS system must keep confidentiality to prevent sensitive name mapping records from being reached by unauthorized data requesters. This is more required in IoT environments where a lot of sensitive data is produced.

The NRS system must also keep confidentiality of meta-data as well as NRS usage to protect the privacy of the users. For instance, a specific user's NRS requests should not be shared outside the NRS system (with the exception of legal intercept).

4.9.2. Authentication

- * NRS server authentication: Authentication of the new NRS servers/nodes that want to be registered with the NRS system must be required so that only authenticated entities can store and update name mapping records. The NRS system should detect an attacker attempting to act as a fake NRS server to cause service disruption or manipulate name mapping records.
- * Producer authentication: The NRS system must support authentication of the content producers to ensure that update/addition/removal of name mapping records requested by content producers are actually valid and that content producers are authorized to modify (or revoke) these records or add new records.
- * Mapping record authentication: The NRS should verify new mapping records that are being registered so that it cannot be polluted with falsified information or invalid records.

4.9.3. Integrity

The NRS system must be prevented from malicious users attempting to hijack or corrupt the name mapping records.

4.9.4. Resiliency and availability

The NRS system should be resilient against denial of service attacks and other common attacks to isolate the impact of the attacks and prevent collateral damage to the entire system. Therefore, if a part of the NRS system fails, the failure should only affect a local domain. And fast recovery mechanisms need to be in place to bring the service back to normal.

5. Conclusion

ICN routing may comprise three steps: name resolution, content request routing, and content delivery. This document investigates the name resolution step, which is the first and most important to be achieved for ICN routing to be successful. A Name Resolution Service (NRS) in ICN is defined as the service that provides such a function of name resolution for translating an object name into some other information such as a locator, another name, metadata, next hop info, etc. that is used for forwarding the object request.

This document classifies and analyzes the NRS approaches according to whether the name resolution step is separated from the content request routing as an explicit process or not. This document also explains the NRS functions used to support heterogeneous name types, producer mobility, scalable routing system, off-path caching, nameless object, manifest, and metadata. Finally, this document presents design considerations for NRS in ICN, which include resolution response time and accuracy, resolution guarantee, resolution fairness, scalability, manageability, deployed system, and fault tolerance.

6. IANA Considerations

There are no IANA considerations related to this document.

7. Security Considerations

A discussion of security guidelines was provided in section 4.9.

8. References

8.1. Normative References

- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlis, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.

8.2. Informative References

- [Ahlgren] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and B. Ohlman, "A Survey of Information-Centric Networking", IEEE Communications Magazine Vol.50, Issue 7, 2012.

- [Xylomenos] Xylomenos, G., Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilako, X., Katsaros, K. V., and G. C. Polyzos, "A Survey of Information-Centric Networking Research, Communications Surveys and Tutorials", IEEE Communications Surveys and Tutorials vol. 16, no. 2, 2014.
- [Baccelli] Baccelli, E., Mehli, C., Hahm, O., Schmidt, T., and M. Wahlsch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild", ACM ICN 2014, 2014.
- [Amadeo] Amadeo, M., Campolo, C., Iera, A., and A. Molinaro, "Named data networking for IoT: An architectural perspective", European Conference on Networks and Communications (EuCNC) , 2014.
- [Quevedo] Quevedo, J., Corujo, D., and R. Aguiar, "A case for ICN usage in IoT environments", IEEE GLOBECOM , 2014.
- [Amadeo2] Amadeo, M. et al., "Information-centric networking for the internet of things: challenges and opportunities", IEEE Network vol. 30, no. 2, July 2016.
- [ID.Zhang2] Ravindran, R. et al., "Design Considerations for Applying ICN to IoT", draft-zhang-icnrg-icniot-03 , May 2019.
- [Koponen] Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K. H., Shenker, S., and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture", ACM SIGCOMM 2007 pp. 181-192, 2007.
- [PURSUIT] "FP7 PURSUIT project.", <http://www.fp7-pursuit.eu/PursuitWeb/> .
- [SAIL] "FP7 SAIL project.", <http://www.sail-project.eu/> .
- [NDN] "NSF Named Data Networking project.", <http://www.named-data.net> .
- [CCNx] "Content Centric Networking project.", <https://wiki.fd.io/view/Cicn> .
- [MF] "NSF Mobility First project.", <http://mobilityfirst.winlab.rutgers.edu/> .
- [Jung] Jung, H. et al., "IDNet: Beyond All-IP Network", ETRI Journal vol. 37, no. 5, October 2015.

- [SA2-5GLAN] 3gpp-5glan, "SP-181129, Work Item Description, Vertical_LAN(SA2), 5GS Enhanced Support of Vertical and LAN Services", 3GPP , http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_82/Docs/SP-181120.zip.
- [Bari] Bari, M. F., Chowdhury, S. R., Ahmed, R., Boutaba, R., and B. Mathieu, "A Survey of Naming and Routing in Information-Centric Networks", IEEE Communications Magazine Vol. 50, No. 12, P.44-53, 2012.
- [Westphal] Westphal, C. and E. Demirors, "An IP-based Manifest Architecture for ICN", ACM ICN , 2015.
- [Mosko] Mosko, M., Scott, G., Solis, I., and C. Wood, "CCNx Manifest Specification", draft-wood-icnrg-ccnxmanifests-00 , July 2015.
- [RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC6920, DOI 10.17487/RFC6920, <https://rfc-editor.org/rfc/rfc6920.txt> , April 2013.
- [Zhang2] Zhang, Y. et al., "A Survey of Mobility Support in Named Data Networking", IEEE Conference on Computer Communications Workshops , 2016.
- [Dannewitz] Dannewitz, C. et al., "Network of Information (NetInf)-An information centric networking architecture", Computer Communications vol. 36, no. 7, April 2013.
- [Ravindran] Ravindran, R. et al., "Forwarding-Label support in CCN Protocol", draft-ravi-icnrg-ccn-forwarding-label-02 , March 2018.
- [Afanasyev] Afanasyev, A. et al., "SNAMP: Secure Namespace Mapping to Scale NDN Forwarding", IEEE Global Internet Symposium , April 2015.
- [Mosko2] Mosko, M., "Nameless Objects", IRTF ICNRG , January 2016.
- [Bayhan] Bayhan, S. et al., "On Content Indexing for Off-Path Caching in Information-Centric Networks", ACM ICN , September 2016.

- [FLIC] Tschudin, C. et al., "File-Like ICN Collection (FLIC)", draft-irtf-icnrg-flic-02 , November 2019.
- [NEO] Eriksson, A. and A. M. Malik, "A DNS-based information-centric network architecture open to multiple protocols for transfer of data objects", 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp. 1-8, 2018.
- [NRSarch] Hong, J. et al., "Architectural Considerations of ICN using Name Resolution Service", draft-irtf-icnrg-nrsarch-considerations-06 , February 2021.
- [RFC9064] Oran, D., "Considerations in the development of a QoS Architecture for CCNx-like ICN protocols", RFC9064, <https://datatracker.ietf.org/doc/rfc9064/> , June 2021.

Acknowledgements

The authors would like to thank Dave Oran, Dirk Kutscher, Ved Kafle, Vincent Roca, Marie-Jose Montpetit, Stephen Farrell, Mirja Kuhlewind, and Colin Perkins for very useful reviews, comments and improvement on the document.

Authors' Addresses

Jungha Hong
ETRI
218 Gajeong-ro, Yuseung-Gu
Daejeon

Email: jhong@etri.re.kr

Tae-Wan You
ETRI
218 Gajeong-ro, Yuseung-Gu
Daejeon

Email: twyou@etri.re.kr

Lijun Dong
Futurewei Technologies Inc.
10180 Telesis Court
San Diego, CA 92121
United States of America

Email: lijun.dong@futurewei.com

Cedric Westphal
Futurewei Technologies Inc.
2330 Central Expressway
Santa Clara, CA 95050
United States of America

Email: cedric.westphal@futurewei.com

Borje Ohlman
Ericsson Research
S-16480 Stockholm
Sweden

Email: Borje.Ohlman@ericsson.com

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: 18 June 2022

J. Hong
T. You
ETRI
V. Kafle
NICT
15 December 2021

Architectural Considerations of ICN using Name Resolution Service
draft-irtf-icnrg-nrsarch-considerations-07

Abstract

This document describes architectural considerations and implications related to the use of a Name Resolution Service (NRS) in Information-Centric Networking (ICN). It explains how the ICN architecture can change when an NRS is utilized and how its use influences the ICN routing system. This document is a product of the Information-Centric Networking Research Group (ICNRG).

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 <https://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 18 June 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Background	4
4. Implications of NRS in ICN	5
5. ICN Architectural Considerations for NRS	6
5.1. Name mapping records registration, resolution, and update	6
5.2. Protocols and Semantics	8
5.3. Routing System	9
6. Conclusion	9
7. IANA Considerations	10
8. Security Considerations	10
9. References	11
9.1. Normative References	11
9.2. Informative References	11
Acknowledgements	13
Authors' Addresses	13

1. Introduction

Information-Centric Networking (ICN) is an approach to evolving the Internet infrastructure to provide direct access to Named Data Objects (NDOs) by names. In two common ICN architectures, NDN [NDN] and CCNx [CCNx], the name of an NDO is used directly to route a request to retrieve the data object. Such direct name-based routing has inherent challenges in enabling a globally scalable routing system, accommodating producer mobility, and supporting off-path caching. These specific issues are discussed in detail in Section 3. In order to address these challenges, a Name Resolution Service (NRS) has been utilized in the literature as well as the proposals of several ICN projects [Afanasyev] [Zhang2] [Ravindran] [SAIL] [MF] [Bayhan].

This document describes the potential changes in the ICN architecture caused by the introduction of NRS and the corresponding implication to the ICN routing system. It also describes ICN architectural considerations for the integration of an NRS. The scope of this

document includes considerations from the perspective of ICN architecture and routing system when using an NRS in ICN. A description of the NRS itself is provided in the companion NRS design considerations document [RFC9138], which provides the NRS approaches, functions and design considerations.

This document represents the consensus of the Information-Centric Networking Research Group (ICNRG). It has been reviewed extensively by the Research Group (RG) members who are actively involved in the research and development of the technology covered by this document. It is not an IETF product and is not a standard.

2. Terminology

- * **Name Resolution Service (NRS):** An NRS in ICN is defined as a service that provides the function of translating a content name or a data object name into some other information such as a routable prefix, a locator, an off-path-cache pointer, or an alias name that is more amenable than the input name to forwarding the object request toward the target destination storing the NDO [RFC9138]. An NRS is most likely implemented through the use of a distributed mapping database system. Domain Name System (DNS) may be used as an NRS. However, in this case, the requirements of frequent updates of NRS records due to the creations of a lot of new NDOs and changes in their locations in the network need to be considered.
- * **NRS server:** An NRS comprises the distributed NRS servers storing the mapping records in their databases. NRS servers store and maintain the mapping records that keep the mappings of content or object name to some other information that is used for forwarding the content request or the content itself.
- * **NRS resolver:** The client side function of an NRS is called an NRS resolver. The NRS resolver is responsible for initiating name resolution request queries that ultimately lead to a name resolution of the target data objects. NRS resolvers can be located in the consumer (or client) nodes and/or ICN routers. An NRS resolver may also cache the mapping records obtained through the name resolution for later usage.
- * **Name registration:** In order to populate the NRS, the content names and their mapping records must be registered in the NRS system by a publisher who has access right to at least one authoritative NRS server or by a producer who generates named data objects. The records contain the mapping of an object name to some information such as other alias names, routable prefixes and locators, which are used for forwarding the content request. Thus, a publisher or

producer of contents creates an NRS registration request and sends it to an NRS server. On registration, the NRS server stores (or updates) the name mapping record in the database and sends an acknowledgement back to the producer or publisher that made the registration request.

- * Name resolution: Name resolution is the main function of the NRS system. It is performed by an NRS resolver which can be deployed on a consumer node or an ICN router. Resolvers are responsible for either returning a cached mapping record (whose lifetime has not expired or alternatively sending a name resolution request toward an NRS server. The NRS server searches for the content name in its mapping record database, and if found, retrieves the mapping record and returns it in a name resolution response message to the NRS resolver.
- * NRS node: NRS servers are also referred to as NRS nodes that maintain the name records. The terms are used interchangeably.
- * NRS client: A node that uses the NRS is called an NRS client. Any node that initiates a name registration, resolution, or update procedure is an NRS client. That is, NRS resolvers, ICN client nodes, ICN routers, or producers can be NRS clients.

3. Background

A pure name-based routing approach in ICN has inherent challenges in enabling a globally scalable routing system, accommodating producer mobility, and supporting off-path caching. In order to address these challenges, an NRS has been utilized in proposals and literature of several ICN projects as follows:

- * Routing scalability: In ICN, application names identifying contents are intended to be used directly for packet delivery, so ICN routers run a name-based routing protocol to build name-based routing and forwarding tables. Similar to the scalability challenge of IP routing, if non-aggregatable name prefixes are injected into the Default Route Free Zone (DFZ) of ICN routers, they would be driving the uncontrolled growth of the DFZ routing table size. Thus, providing the level of indirection enabled by an NRS in ICN can be an approach to keeping the routing table size under control. The NRS system resolves name prefixes which do not exist in the DFZ forwarding table into globally routable prefixes such as one proposed in NDN [Afanasyev]. Another approach dealing with routing scalability is the Multi-level Distributed Hash Table (MDHT) used in NetInf [Dannewitz]. It provides name-based anycast routing that can support a non-hierarchical namespace and can be adopted on a global scale [Dannewitz2].

- * **Producer mobility:** In ICN, if a producer moves into a different name domain that uses a different name prefix, the request for a content produced by the moving producer with the origin content name must be forwarded to the moving producer's new location. Especially in a hierarchical naming scheme, producer mobility support is much harder than in a flat naming scheme since the routing tables in a broader area need to be updated to track the producer movement. Therefore, various ICN architectures such as NetInf [Dannewitz] and MobilityFirst [MF] have adopted NRS systems to tackle the issues of producers whose location changes.
- * **Off-path caching:** In-network caching is a common feature of an ICN architecture. Caching approaches can be categorized into on-path caching and off-path caching, according to the location of caches in relation to the forwarding path from the original content store to a consumer. Off-path caching, sometimes also referred to as content replication or content storing, aims to replicate a Named Data Object in various locations within a network in order to increase the availability of contents, reduce access latency, or both. These caching locations may not be lying along the content forwarding path. Thus, finding off-path cached contents requires more complex forwarding procedures if a pure name-based routing is employed. In order to support access to off-path caches, the locations of replicas are usually advertised into a name-based routing system or into an NRS as described in [Bayhan].

This document discusses architectural considerations and implications of ICN when an NRS is utilized to solve such challenges facing a name-based routing in ICN.

4. Implications of NRS in ICN

An NRS is not a mandatory part of an ICN architecture, as the majority of ICN architectures uses name-based routing that avoids the need for a name resolution procedure. Therefore, the utilization of an NRS in an ICN architecture changes some architectural aspects at least with respect to forwarding procedures, latency, and security, as discussed below:

- * **Forwarding procedure:** When an NRS is included in an ICN architecture, the name resolution procedure has to be included in the ICN overall routing and forwarding architectural procedures. To integrate an NRS into an ICN architecture, there are certain things that have to be decided and specified such as where, when and how the name resolution task is performed.

- * Latency: When an NRS is included in an ICN architecture, additional latency introduced by the name resolution process is incurred by the routing and forwarding system. Although the latency due to the name resolution is added, the total latency of individual requests being served could be lower if the nearest copies or off-path caches can be located by the NRS lookup procedure. Additionally, there might be a favorable trade-off between the name resolution latency and inter-domain traffic reduction by finding the nearest off-path cached copy of the content. Finding the nearest cache holding the content might significantly reduce the content discovery as well as delivery latency.
- * Security: When any new component such as an NRS is introduced in the ICN architecture, the attack surface may increase. Protection of the NRS system itself against attacks such as Distributed Denial of Service (DDoS) and spoofing or alteration of name mapping records and related signaling messages may be challenging.

5. ICN Architectural Considerations for NRS

This section discusses the various items that can be considered from the perspective of ICN architecture when employing an NRS system. These items are related to the registration, resolution, and update of name mapping records, protocols and messages, and integration with the routing system.

5.1. Name mapping records registration, resolution, and update

When an NRS is integrated in ICN architecture, the functions related to the registration, resolution and update of name mapping records have to be considered. The NRS nodes maintain the name mapping records and may exist as an overlay network over the ICN routers, that is, they communicate to each other through ICN routers. Figure 1 shows the NRS nodes and NRS clients connected through an underlying network. The NRS nodes should be deployed in such a manner that an NRS node is always available at a short distance from an NRS client so that communication latency for the name registration and resolution requested by the NRS client remains very low. The name registration, name resolution and name record update procedures are briefly discussed below.

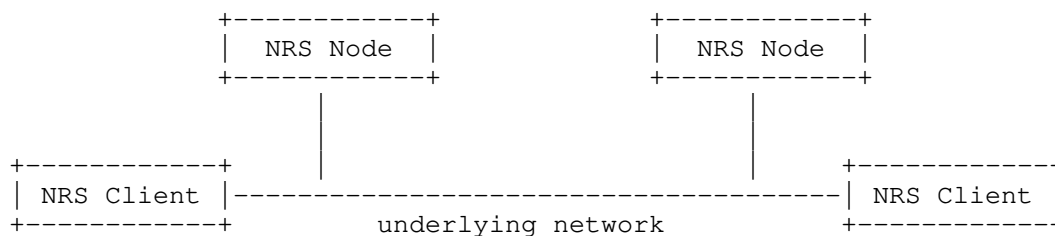


Figure 1: NRS nodes and NRS clients connected through an underlying network

- * Name registration: Name registration is performed by the producer (as an NRS client) when it creates a new content. When a producer creates content and assigns a name from its name prefix space to the content, the producer performs the name registration in an NRS node. Name registration may be performed by an ICN router when the ICN architecture supports off-path caching or cooperative caching since involving an NRS may be a good idea for off-path caching. The ICN routers with forwarder caches do not require name registration for their cached content because they lie on the path toward an upstream content store or producer. They will be hit when a future request is forwarded to the content producer by an ICN router lying downstream toward the ICN client node. However, ICN routers performing off-path caching of content must invoke the name registration procedure so that other ICN routers can depend on name resolutions to know about the off-path cache locations. If a content gets cached in many off-path ICN routers, all of them may register the same content names in the same NRS node, resulting in multiple registration actions. In this case, the NRS node adds the new location of the content to the name record together with the previous locations. In this way, each of the name records stored in the NRS node may contain multiple locations of the content. Assigning validity time or a lifetime of each mapping record may be considered especially for the off-path caching contents and managing mobility.
- * Name resolution: Name resolution is performed by an NRS client to obtain the name record from an NRS node by sending a name resolution request message and getting a response containing the record. In the name-based ICN routing context, the name resolution is needed by any ICN router whose forwarding information base (FIB) does not contain the requested name prefix. Name resolution may also be performed by the consumer (especially in the case where the consumer is multihomed) to forward the content request in a better direction so that it obtains the

content from the nearest cache. If the consumer is single homed, it may not bother to perform name resolution, instead depending on either straightforward name-based routing or name resolution by an upstream ICN router. On this case, the consumer creates the content request packet containing the content name and forwards to the nearest ICN router. The ICN router checks its FIB table to see where to forward the content request. If the ICN router fails to know the requested content reachable, it performs name resolution to obtain the name mapping record and adds this information to its FIB. The ICN router may also perform name resolution even before the arrival of a content request to use the name mapping record to configure its FIB.

- * Name record update: Name record update is carried out when a content name mapping record changes, e.g. the content is not available in one or more of previous locations. The name record update includes the substitution and deletion of the name mapping records. The name record update may take place explicitly by the exchange of name record update messages or implicitly when a timeout occurs and a name record is deemed to be invalid. The implicit update is possible when each record is accompanied by a lifetime value. The lifetime can be renewed only by the authoritative producer or node. The cached mapping records get erased after the lifetime expires unless a lifetime extension indication is obtained from the authoritative producer.

5.2. Protocols and Semantics

In order to develop an NRS system within a local ICN network domain or global ICN network domain, new protocols and semantics must be designed and implemented to manage and resolve names among different name spaces.

One way of implementing an NRS for CCNx is by extending the basic TLV format and semantics [RFC8569] [RFC8609]. For instance, name resolution and response messages can be implemented by defining new type fields in the Interest and Content Object messages [CCNxNRS]. By leveraging the existing CCNx Interest and Content Object packets for name resolution and registration, the NRS system can be deployed with a few ICN protocol changes. However, because of confining the changes to the basic ICN protocol and semantics, the NRS system may not be able to exploit more flexible and scalable designs.

On the other hand, an NRS system can be designed independently with its own protocol and semantics like the NRS system described in [Hong]. For instance, the NRS protocol and messages can be implemented by using a RESTful API and the NRS can be operated as an application protocol independent of the rest of the ICN protocol.

5.3. Routing System

An NRS reduces the routing complexity of ICN architecture compared to pure name-based routing. It does so by permitting the routing system to update the routing table on demand through the help of name records obtained from NRS. The routing system therefore needs to make name resolution requests and process the information returned, such as a prefix, a locator, an off-path-cache pointer, or an alias name, obtained from the name resolution.

No matter what kind of information is obtained from the name resolution, as long as it is in the form of a name, the content request message in the routing system may be reformatted with the obtained information. In this case, the content name requested originally by a consumer needs to be involved in the reformatted content request to check the integrity of the binding between the name and the requested content. In other words, the information obtained from the name resolution is used to forward the content request and the original content name requested by a consumer is used to identify the content. Alternatively, the resolved information may be used to build the routing table.

The information obtained from name resolution may not be in the form of a name. For example, it may identify tunnel endpoints by IP address and instead be used to construct an IP protocol tunnel through which to forward the content request.

6. Conclusion

A Name Resolution Service (NRS) is not a mandatory part in an ICN architecture, as the majority of ICN architectures use name-based routing which does not employ a name resolution procedure. However, such name-based routing in ICN has inherent challenges in enabling a globally scalable routing system, accommodating producer mobility, and supporting off-path caching. In order to address these challenges, an NRS system has been introduced in several ICN projects. Therefore, this document describes how the ICN architecture changes when an NRS is utilized and how this affects the ICN routing system.

The document defines a few terminologies related to an NRS and explains some inherent challenges of pure name-based routing in ICN such as routing scalability, producer mobility, and off-path caching. This document describes how the ICN architecture would change with respect to procedures, latency, and security when an NRS is utilized. According to the ICN architectural changes, this document describes ICN architectural considerations for NRS such as the functions related to the registration, resolution and update of name mapping records, protocols and semantics to implement an NRS system, and the routing system involving the name resolution.

7. IANA Considerations

There are no IANA actions required by this document.

8. Security Considerations

When any new component such as an NRS is introduced in the ICN architecture, the attack surface increases. The related security vulnerability issues are discussed below:

- * Name space security: In order to deploy an NRS system in ICN architecture, ICN name spaces, which may be assigned by authoritative entities, must be securely mapped to the content publishers and securely managed by them. According to the ICN research challenges [RFC7927], a new name space can also provide an integrity verification function to authenticate its publisher. The issues of namespace authentication and the mapping among different name spaces require further investigation.
- * NRS system security: An NRS requires the deployment of new entities (e.g. NRS servers) to build distributed and scalable NRS systems. Thus, the entities, e.g., NRS server maintaining a mapping database, could be the focus of attacks by receiving malicious requests from innumerable adversaries comprising a Denial of Service or Distributed Denial of service attacks. In addition, NRS clients in general must trust the NRS nodes in other network domains to some degree and communication among them must also be protected securely to prevent malicious entities from participating in this communication. The history of name resolution data requires to be stored and analyzed as in passive DNS to uncover potential security incidents or discover malicious infrastructures.
- * NRS protocol and message security: In an NRS system, the protocols to generate, transmit and receive NRS messages related to the name registration, resolution, and record update should be protected by proper security mechanisms. Proper security measures must be

provided so that only legitimate nodes can initiate and read NRS messages. These messages must have secured by integrity protection and authentication mechanism so that unauthorized parties cannot manipulate them when being forwarded through the network. Security measures to encrypt these messages should also be developed to thwart all threats to both security and privacy. Numerous problems similar to the security issues of an IP network and DNS can affect the overall ICN architecture. The DNS QNAME minimization type of approach would be suitable for preserving privacy in the name resolution process. Therefore, security mechanisms such as accessibility, authentication, etc., for the NRS system [RFC9138] should be considered to protect not only the NRS system, but also the ICN architecture overall.

9. References

9.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", <https://www.rfc-editor.org/info/rfc8569> , July 2019.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", <https://www.rfc-editor.org/info/rfc8609> , July 2019.
- [RFC9138] Hong, J. et al., "Design Considerations for Name Resolution Service in Information-Centric Networking (ICN)", <https://www.rfc-editor.org/info/rfc9138> , November 2021.

9.2. Informative References

- [NDN] "NSF Named Data Networking project.", <http://www.named-data.net> .

- [CCNx] "Content Centric Networking project.",
<https://wiki.fd.io/view/Cicn> .
- [Afanasyev] Afanasyev, A. et al., "SNAMP: Secure Namespace Mapping to Scale NDN Forwarding", IEEE Global Internet Symposium , April 2015.
- [Zhang2] Zhang, Y., "A Survey of Mobility Support in Named Data Networking", NAMED-ORIENTED MOBILITY: ARCHITECTURES, ALGORITHMS, AND APPLICATIONS (NOM) , 2016.
- [Ravindran] Ravindran, R. et al., "Forwarding-Label support in CCN Protocol", draft-ravi-icnrg-ccn-forwarding-label-01 , July 2017.
- [SAIL] "FP7 SAIL project.", <http://www.sail-project.eu/> .
- [MF] "NSF Mobility First project.",
<http://mobilityfirst.winlab.rutgers.edu/> .
- [Bayhan] Bayhan, S. et al., "On Content Indexing for Off-Path Caching in Information-Centric Networks", ACM ICN , September 2016.
- [CCNxNRS] Hong, J. et al., "CCNx Extension for Name Resolution Service", draft-hong-icnrg-ccnx-nrs-02 , July 2018.
- [Hong] Hong, J., Chun, W., and H. Jung, "Demonstrating a Scalable Name Resolution System for Information-Centric Networking", ACM ICN , September 2015.
- [Dannewitz] Dannewitz, C. et al., "Network of Information (NetInf)-An information centric networking architecture", Computer Communications vol. 36, no. 7, April 2013.
- [Dannewitz2] Dannewitz, C., D'Ambrosio, M., and V. Vercellone, "Hierarchical DHT-based name resolution for Information-Centric Networks", Computer Communications vol. 36, no. 7, April 2013.

Acknowledgements

The authors would like to thank Dave Oran (ICNRG Co-chair) for very useful reviews and comments on the document and they helped to immeasurably improve the document.

Authors' Addresses

Jungha Hong
ETRI
218 Gajeong-ro, Yuseung-Gu
Daejeon

Email: jhong@etri.re.kr

Tae-Wan You
ETRI
218 Gajeong-ro, Yuseung-Gu
Daejeon

Email: twyou@etri.re.kr

Ved Kafle
NICT
4-2-1 Nukui-Kitamachi, Tokyo
184-8795
Japan

Email: kafle@nict.go.jp

Information-Centric Networking Research Group
Internet-Draft
Intended status: Informational
Expires: May 19, 2020

R. Li
H. Asaeda
NICT
November 16, 2019

Hop-by-Hop Authentication in Content-Centric Networking/Named Data
Networking
draft-li-icnrg-hopauth-01

Abstract

The unpredictability of consumers, routers, copyholders, and publishers for the in-network data retrievals in Content-Centric Networking (CCN) / Named Data Networking (NDN) poses a challenge to design an authentication mechanism to inhibit the malicious consumers to flood data requests and prevent the fake data from being provided. Signature is adopted as the fundamental function in CCN / NDN, which however can only provide publisher authentication with additional certificate acquisition. This document describes the Hop-by-Hop Authentication mechanism (HopAuth) integrating certificate collection and packet forwarding potentially with the assistance from certificate authority to provide consumer authentication, copyholder authentication and path authentication to enable the in-network data retrieval to be trustworthy, besides the publisher authentication.

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 <https://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 May 19, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. System Descriptions	5
4. HopAuth Designs	7
4.1. Initial Trust Establishment	7
4.2. Data-centric Certificate Management	8
4.2.1. Certificate Exchange	8
4.2.2. Certificate Update and Revocation	9
4.3. Forwarding-Integrated Authenticable Data Retrieval	10
4.4. Suspension-Chain Model (SCM)	11
5. Protocol Message Format	12
6. Security Considerations	13
7. References	13
7.1. Normative References	13
7.2. Informative References	13
Authors' Addresses	15

1. Introduction

Information-Centric Networks in general, and Content-Centric Networking (CCN) [3] or Named Data Networking (NDN) [4] in particular, are the emerging network architectures enabling in-network caching and data retrievals through their names. In CCN/NDN, data can be cached at the intermediate routers, close to consumers for reducing delay and redundant bandwidth consumption or for the robustness under dynamic network environment. It has been noticed that CCN/NDN is a promising approach for the application scenarios in disaster networking [5], video streaming [6], and Internet of Things (IoT) [8].

In CCN/NDN, the basic network operations and these use scenarios with in-network data caching and retrievals lead the network to be seriously vulnerable under a variety of attacks, such as the impersonation attack, malicious-request attack [9], [10], [11], and the data poisoning attack [12], [13], [14]. The unpredictability of consumers, routers, copy holders, and publishers during data

retrievals in CCN/NDN poses the novel challenge to design data-centric authentication to prevent these attacks. This novel authentication should potentially enable the consumer authentication, copyholder authentication to authenticate the identity of the entity providing data, path authentication to authenticate the path from which data are retrieved, in addition to the publisher authentication.

On the other hand, signature is already adopted as the fundamental function in CCN/NDN, which promises to achieve the integrity and publisher authentication. It can partially prevent the above attacks and but still is insufficient to protect the unpredictable data retrievals in CCN/NDN. The unpredictability with which copy holders provide data, routers cache data, and consumers request data leads to great difficulty in inhibiting malicious-request attacks and data-poisoning attacks. To prevent data poisoning, consumers and routers need to verify data before caching them. If the data are found to be fake, the copy holder providing the data and the path to retrieve the data also need to be discovered in order to disable the further spread of that fake data. To prevent malicious-request attacks, copy holders need to verify the identities of the consumers.

There are many authentication mechanisms with key management schemes in the Internet, such as Kerberos [15], MSEC [16], X.509 [17], PGP [18], RPKI [19]. They are designed to achieve different purposes with centralized or decentralized approach based on end-to-end communication paradigm in the Internet. They can only provide the authentications of consumers and publishers or the link between them. In other words, they do not consider data-centric authentication, that is aimed to protect data in the networks.

For data-centric authentication, the key is to protect or prevent data-poisoning and malicious-request attacks. Regarding the data poisoning attacks, if the fake or corrupted data are cached along the path, consumers who use the path always retrieve the wrong data, because the router does not detect the cached data validity (as it is signed by attacker correctly). The fake or corrupted data are further cached, which pollute the routers as virus spreads. These routers will provide the fake or corrupted data to other potential consumers. Therefore, to inhibit the data poisoning attacks, routers need to verify the data before caching them and consumers need to verify the copyholder and the path to retrieve data.

For malicious requests, even malicious Interests can be reduced by restricting sending rate, part of malicious Interests still can reach the copyholder. If copyholders verify the Interests before replying the data, the effect to inhibit the Interest flooding attack can be improved. Hence, the authentication from any consumer or router to

Interest, data, copyholder, and the data retrieval path plays crucial role to inhibit the data poisoning and malicious-request attacks, besides the identity authentication to the publisher.

Another concern is that most of the existing authentication mechanisms rely on centralized servers to acquire keys or certificates, thereby increasing authentication delays, which we refer to herein as the delay-enlargement problem. Obviously, they cannot satisfy the following performance requirements for the emerging data-centric communication paradigm in CCN/NDN: any consumer or router needs to authenticate Interest, data, copyholder (including publisher), and the data retrieval path without additional messaging delay.

In this document, we describe a secure mechanism named HopAuth, where forwarding-integrated hop-by-hop certificate collection is performed together with the adaptive replacement for parts of chain with highly trustworthy certificates. This specification derives directly from the design published in [20]. In HopAuth, a suspension-chain model (SCM) is used as the trust model, where the neighbor-trust-based certificate chain is suspended by certificate authority (CA)-based trust. The HopAuth avoids reliance on centralized server(s) for chain construction and solves the delay-enlargement problem.

2. Terminology

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 [1].

The following terminology is used throughout this document.

- o Cryptographic key: A string of bits used by a cryptographic algorithm to transform plain-text into cipher-text or vice versa.
- o Signature: A cryptographic value calculated through public key algorithm from the data and a secret key only known by the signer. It is to validate the authenticity and integrity of a message.
- o Certificate: A data structure used to verifiably bind an identity to a cryptographic key.
- o Consumer: A node requesting data. It initiates communication by sending an interest packets.
- o Publisher: A node providing data. It originally creates or owns the data.

- o Router: A node forwarding data. It may hold memory to cache the data.
- o Forwarding Information Base (FIB): A lookup table in a router containing the name prefix and corresponding destination interface to forward the interest packets.
- o Pending Interest Table (PIT): A lookup table populated by the interest packets containing the name prefix of the requested data, and the outgoing interface used to forward the received data packets.
- o Content Store (CS): A storage space for a router to cache data objects. It is also known as in-network cache.
- o Physical entity: an entity that communicates using a physical device. This could be a router or a publisher node (PN) that hosts applications.
- o Logical entity: an entity that is involved in an application. This can be an authorizer, a sub-authorizer, or a publisher.

3. System Descriptions

Here we briefly explain the background and basic network operations of CCN/NDN. Different from the end-to-end communications in the Internet, CCN/NDN provides data-name-based retrievals as in Fig. 1. It further requires the data-centric authentication, instead of the current end-to-end secure channel establishment.

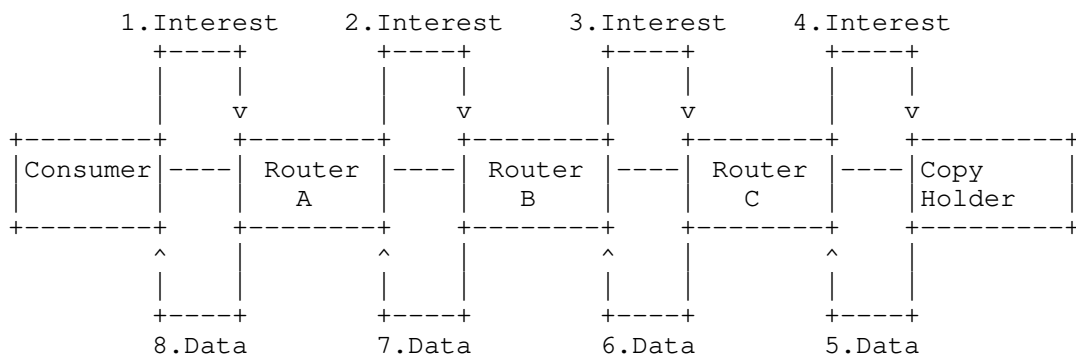


Figure 1: Request and reply messages forwarded by consumer, copy holder and routers.

In a CCN/NDN network, each router in a CCN/NDN network has three main data structures: a FIB for forwarding Interests, a CS for caching data, and a PIT for forwarding data. Basically there are two types of packets: interest and data. As in Fig. 1, consumer requests data by throwing an "interest" packet with the name of data to the network. Regarding the difference to note here between CCN [3] and NDN [4] is that in later versions of CCN, interest packet must carry a full data name, while in NDN it may carry a data name prefix.

Once a router receives an "interest" packet, it performs a series of the following look-up.

The router first checks in the CS to see whether it holds the corresponding data or not. If there is, it returns the data through the reverse path for forwarding interest packet as the copy holder in Fig. 1. If not, it performs a look-up of the PIT. If there is already a PIT entry matching the name of requested data, it is updated with the incoming interface of this new request and the interest is discarded. If there is no matching entry, it creates an entry in the PIT that lists the data name and the interfaces from which it received the interest. Then, the interest undergoes a FIB lookup to discover the outgoing interface.

Once a copy of the "data" packet is retrieved, the router sends it back to the data requester(s) using the trail of PIT entries and remove the PIT state every time that an interest is satisfied. Additionally, it may store the data in its CS.

However, data retrieval with in-network caching in CCN/NDN has been identified to suffer from malicious data-request attacks [9], [10], [11], and the data poisoning attacks [12], [13], [14]. In the former, adversaries impersonate consumers to create a flood of interests, and in the latter, they impersonate copy holders (e.g., routers or publishers) to provide fake data. These attacks are severe, because data are cached in a distributed manner, and copy holders have no way to verify consumers' identities, consumers/routers have no way to verify copy holders' identities to avoid caching fake data, and the path to retrieve data cannot be verified. This form of attack can quickly pollute the router caches as the virus spreads, because routers cache the fake data, redistribute them, and other intermediate routers re-cache them. It finally consumes much in-network caches and prevents consumers from retrieving the correct data.

4. HopAuth Designs

Herein we elaborate the basic design of HopAuth, which has three phases: 1) initial trust-establishment, 2) data-centric certificate management, and 3) forwarding-integrated authenticable data-retrieval. The trust model for HopAuth is given by a suspension chain model (SCM) later explained.

For the first phase, certificates are issued for neighboring entities and the certificate authority (CA) potentially issues certificates to a set of routers to form the highly trusted router group (HTRG). Let $CE(A \rightarrow B)$ represent the public-key certificate issued from A to B. In the second phase, routers exchange certificates within their neighborhoods, and any compromised entity can be shut down quickly. Finally, the third phase provides a hop-by-hop method for constructing a chain consisting of a physical-entity certificate chain (peCEChain) and a logical-entity certificate chain (leCEChain) for data-centric authentication.

4.1. Initial Trust Establishment

The initial trust establishment has two components: self-certifiable naming and certificate issuing.

Self-certifiable naming defines the rule for naming the principals, including entities, keys, and certificates, to enable the entities to be self-certifiable. As the extension of the cryptographically generated address (CGA) [7], we merge the hash-based self-certifying names with hierarchical naming. It embeds the public key into the name, which is self-certifiable. The purpose of self-certifiable naming is to prevent stealing and spoofing of existing names. The public key of the name owner is bound cryptographically to the name. The name owner can use the corresponding private key to assert its ownership and to sign messages sent from the entity with that name. In fact, an attacker can create a new name from an arbitrary public key. However, the attacker cannot impersonate somebody else's name.

To issue a certificate, an entity should be convinced that a given public key truly belongs to another entity. Under that situation, it issues a certificate for this entity such that the public key is bound to the entity name by its signature. In HopAuth, certificate issuing is the initial trust establishment among entities having trust relationships, namely as neighbor-based trust for physical entities, CA-based trust, and authorization relationships for logical entities.

For neighbor-based trust, a physical entity creates a public key and the corresponding private key locally by itself. It generates the

names using the public key based on the self-certifiable naming method. If physical entity B is a neighbor of entity A, B can announce its public key to A with the key name and A generates the certificate for B.

CA-based trust between two entities is established using the CA as the "introducer". The CA is managed by the network operator, and provides certificates to the owner's entities and the highly trusted physical entities close to them in the HTRG. The entities in the HTRG then confer CA-based trust relationships and issue certificates to each other.

All the highly trustable entities register at CA. CA has the whole view of these entities and determines the neighboring relations among the entities in HTRG. After determination of the neighboring relations, an entity, A, sends interest to CA to request the certificates of its neighbors. CA replies with the related certificates to A, such as $CE(CA \rightarrow B)$. After A receiving the certificates, A verifies them, issues certificates from itself to those neighbors, such as $CE(A \rightarrow B)$, and sends to CA. CA verifies this certificate and forwards it to B when B requests.

In HopAuth, the physical entities pre-keep the certificates and associate certificates with interfaces in FIB. The physical entity knows the next hop of one interface, and it associates the certificate from that entity to itself with the forwarding interface. This mechanism enables the appending of the relevant certificate to the packets as required when forwarding them.

For logical entities, the trust relationship is defined by the application. Each logical entity also generates a public/private key pair by itself. If logical entity A authorizes the right for entity B to manage a sub-category or publish data, A should provide a certificate for the true public key of B.

4.2. Data-centric Certificate Management

4.2.1. Certificate Exchange

Certificate exchange allows entities to share the certificates that they issue and hold. Each physical entity has a local repository in which to store certificates securely. In HopAuth, the physical entities request and keep all the certificates issued for or by their nearby highly trusted entities in the HTRG and by common neighboring physical entities. To exchange the information of certificates they hold, each entity hashes the names of certificates and exchange hashes to see if there is one missed in the neighborhood. If there is, the entity will send interest to its neighbor to retrieve that

certificate. Finally, the physical entities hold all the certificates within a two-hop distance and the certificates with nearby highly trusted entities in the HTRG. These certificates are used to construct chains hop by hop, shorten certificate chains, and check the certificates appended by an up-stream entity. This certificate check is performed by the next hop of the entity to check whether the certificate appended by this entity is the same as the one stored by it. If the certificates are the same, the certificate passes the check. Otherwise, this packet will be dropped because of the fake certificate.

The certificates of the logical entities in an application should be stored in the repository of the PN. When data are published, the trust chain from the PN to the publisher can be automatically formed and appended to the packet. Meanwhile, the neighbors of the PN also keep all the certificates of the PN in order to check them during transmissions.

4.2.2. Certificate Update and Revocation

To guarantee its validity, each certificate in the network is issued with a certificate expiration time, after which the certificate is invalid.

For certificate updates using neighbor-based trust, the subject entity of the certificate should notify the issuer of its interest in updating the certificate. On receiving this interest, the issuer checks whether this entity has been compromised. It then checks whether the mapping between the name and the public key satisfies the naming rule. If all the checks are passed, the issuer considers whether the public key of the subject entity is still trustworthy, generates an updated certificate, and replies with this updated certificate. If any check is failed, the issuer does not provide a certificate update to that entity. For certificate updates using CA-based trust in the HTRG, the subject entity of the certificate requests the CA to issue an updated certificate and provide it to the related nearby highly trusted routers, who can further issue update certificates.

If one entity no longer wishes to trust another entity, the former may revoke the certificate that it originally issued. Because certificates are issued over a one-hop distance, it is easy to detect the misbehavior of an entity. To revoke a certificate quickly, the revocation is announced over a two-hop distance. The revocation initiator broadcasts the revocation information to all the entities within a two-hop distance. Each entity receiving this information adds the compromised entity or certificate to its blacklist. All the

packets from the compromised entities are dropped for a rapid local shutdown.

4.3. Forwarding-Integrated Authenticable Data Retrieval

Here, when forwarding interests and data, we define a forwarding-integrated hop-by-hop approach to construct the suspended chain from unpredictable copy holders to a consumer for authenticating interest, and from a consumer or router to data for authenticating copy holders or publishers or path to retrieve data. We let highly trusted routers replace the parts of chain with highly trusted certificates induced by CA's suspended trust to enhance trustworthiness.

The steps for data retrieval are as follows.

Step 1: The consumer issues an interest appended with its signature. It knows its next hop is the router to which it is connected. It then appends to this interest the certificate from the next-hop router to itself. Finally, it sends out the interest.

Step 2: When the interest is received by an router, it checks whether the previous certificates are correct. If the check succeeds and it belongs to the HTRG, this router attempts to find the previous highly trusted router to replace the related part of the certificate chain with one highly trusted certificate in the suspended chain. If this IPE does not belong to the HTRG, it directly finds the interface to the next hop and appends to the interest the relevant certificate from the next-hop router to itself.

Step 3: This is executed if an router holds the requested data in its cache. The router checks the suspended chain from it to the consumer through the process described later in the SCM. If the verification succeeds, this router replies with the data packet. In the data packet, the suspended chain from this router to the publisher and the certificate from the next router to this router are appended. This router sends this data packet to the interface in reply as specified in the PIT.

Step 4: If the PN receives the interest, it verifies the suspended chain from itself to the consumer. If the verification succeeds, the PN discovers the suspended chain from itself to the publisher in its storage, and appends this certificate chain with the certificate from the first router to itself. Next, the PN replies with these data using the reverse path of the interest.

Step 5: After the router receives the data packet, it performs forwarding and possibly caching. When the router intends to cache the data, it first caches them in a temporary cache, which is

separated from the data cache. Second, it checks the suspended chain from itself to the publisher. Only if the verification passes, these data can be cached in the data memory and the suspended chain from this router to the publisher will be cached along with the data. The verification is performed offline, which does not affect the speed of data retrieval. At the same time, this router checks the previous certificate. If the check is successful and it belongs to the HTRG, it will discover the previous entity in the suspended chain belonging to the HTRG. If there is a previous entity, the router replaces part of the related certificate path with a highly trusted certificate. Otherwise, the router directly finds the interface to the corresponding certificate from the next hop to itself and appends this certificate to the packet, then forwards this packet to the interface.

Step 6: After the consumer receives the data packet, there should be a certificate chain from it to the publisher. It verifies this suspended chain. If the verification passes, it believes that it gets the authentic public key of the publisher, and utilizes the key to verify the signature of the data. The consumer can also verify the copy holder or the routers on the path.

4.4. Suspension-Chain Model (SCM)

A suspension-chain model (SCM) is the trust model in HopAuth. It is a flexible series of neighbor-trust-based certificates suspended by CA's trust, which form a suspension chain. In SCM, neighbor-based trust forms the certificate chain to realize data-centric authentication, whereas CA-based trust reduces the length of the chain to solve the trust degradation problem for certificate chain. For CA's trust, the CA assigns certificates to highly trusted routers as the suspension points based on CA's trust, which is the pre-trust between these routers and CA.

Public-key verification is the process for one entity to verify the authenticity of the public key of another entity. Entity X authenticates the public key of the entity Y by verifying the suspension chain in the following steps. First, X verifies the first certificate by its private key. If it is correct, each intermediate public key is used to verify the next direct associated certificate. This process continues for multiple rounds until the final certificate is verified. Finally, X obtains the true public key of the entity Y.

5. Protocol Message Format

HopAuth messages are encoded in the CCNx TLV format [2]. An example of HopAuth packet format with a certificate chain from consumer C0 to router R_n, (CE1(C0→R1), ..., CE_n(R_(n-1)→R_n)), is provided in Figure 2. As in Figure 2, the HopAuth message consists HopAuth header, and HopAuth certificate TLVs.

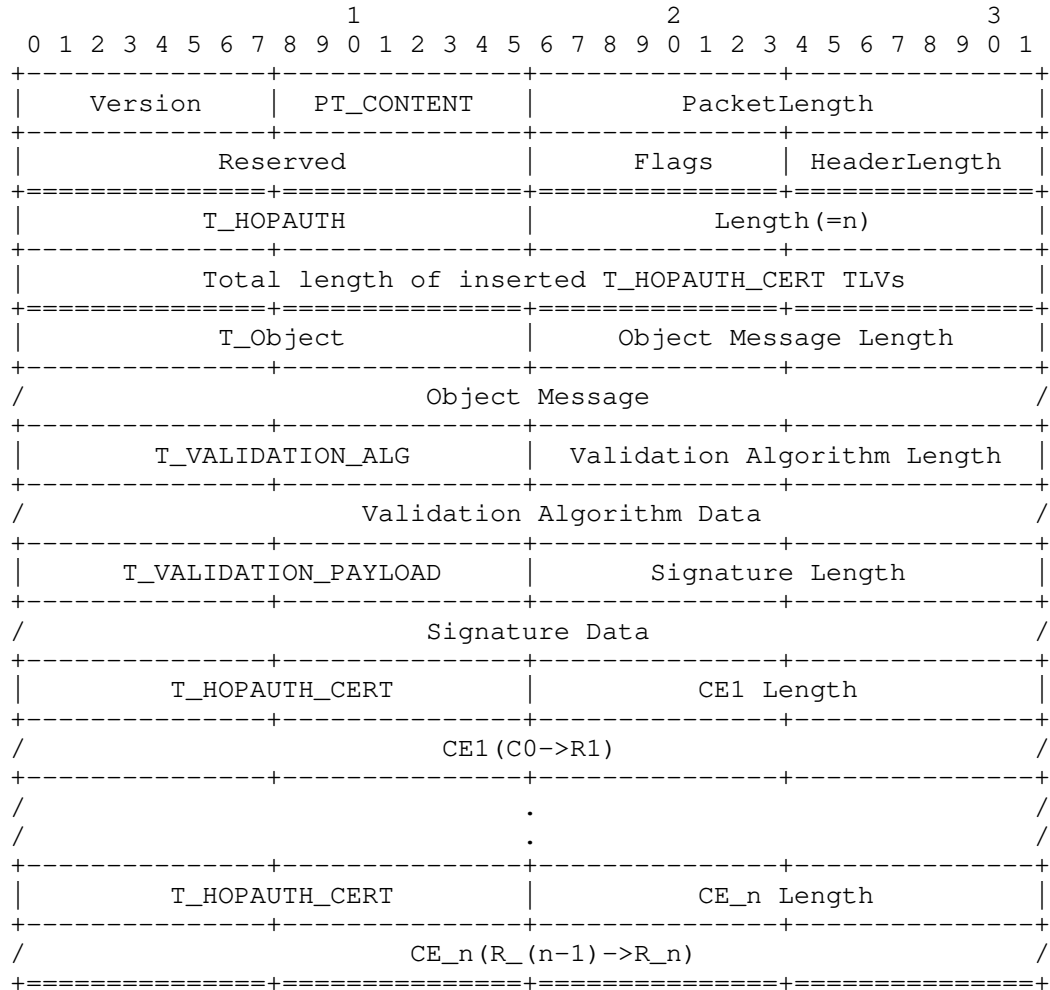


Figure 2: An Example of HopAuth packet format

The HopAuth header is composed of the fields of type, Length, and Total length of inserted T_HOPAUTH_CERT's TLVs. The type is

T_HOPAUTH, the Length shows the number of inserted certificates, and Total length of inserted T_HOPAUTH_CERT's TLVs describes the total number of bits occupied by certificate TLVs. In the part of HopAuth certificate TLVs, the certificate name, length and certificate data are included. Take CE1(C0->R1) in Figure 2 as example. The type is T_HOPAUTH_CERT, CE1 Length is the length of certificate CE1(C0->R1), and CE1(C0->R1) is the certificate data for CE1(C0->R1).

6. Security Considerations

This document is entirely about an authentication mechanism in CCN/NDN.

7. References

7.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [2] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", RFC 8609, July 2019.

7.2. Informative References

- [3] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", Proc. CoNEXT, ACM, December 2009.
- [4] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and B. Zhang, "Named data networking", ACM Comput. Commun. Rev., vol. 44, no. 3, July 2014.
- [5] Seedorf, J., Arumaithurai, M., Tgami, A., Ramakrishnan, K., and N. Melazzi, "Research Directions for Using ICN in Disaster Scenarios", draft-irtf-icnrg-disaster-07 (work in progress), June 2019.
- [6] Westphal, C., Lederer, S., Posch, D., Timmerer, C., Azgin, A., Liu, W., Mueller, C., Detti, A., Corujo, D., Wang, J., Montpetit, M., and N. Murray, "Adaptive Video Streaming over Information-Centric Networking (ICN)", RFC 7933, August 2016.

- [7] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [8] Ravindran, R., Zhang, Y., Grieco, L., Lindgren, A., Burke, J., Ahlgren, B., and A. Azgin, "Design Considerations for Applying ICN to IoT", draft-irtf-icnrg-icniot-03 (work in progress), May 2019.
- [9] Afanasyev, A., Mahadevan, P., Moiseenko, I., Uzun, E., and L. Zhang, "Interest flooding attack and countermeasures in named data networking", Proc. IFIP Networking, IFIP, May 2013.
- [10] Compagno, A., Conti, M., Gasti, P., and G. Tsudik, "Poseidon: mitigating interest flooding ddos attacks in named data networking", Proc. LCN 2013, IEEE, October 2013.
- [11] Nguyen, T., Cogranne, R., and G. Doyen, "An optimal statistical test for robust detection against interest flooding attacks in ccn", Proc. International Symposium on Integrated Network Management (INM), IFIP/IEEE, May 2015.
- [12] Ghali, C., Tsudik, G., and E. Uzun, "Network-layer trust in named-data networking", ACM SIGCOMM Computer Communication Review, vol.44, no. 5, October 2014.
- [13] Kim, D., Nam, S., Bi, J., and I. Yeom, "Efficient content verification in named data networking", Proc. ACM Conference on Information-Centric Networking, ACM, September 2015.
- [14] Gasti, P., Tsudik, G., Uzun, E., and L. Zhang, "Dos and ddos in named data networking", Proc. IEEE ICCCN 2013, IEEE, August 2013.
- [15] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.
- [16] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.
- [17] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

- [18] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.
- [19] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol Version 1", RFC 8210, September 2017.
- [20] Li, R., Asaeda, H., and J. Wu, "DCAuth: Data-Centric Authentication for Secure In-Network Big-Data Retrieval", IEEE Transactions on Network Science and Engineering, DOI: 10.1109/TNSE.2018.2872049, September 2018.

Authors' Addresses

Ruidong Li
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: lrd@nict.go.jp

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

Information-Centric Networking Research Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2020

R. Li
H. Asaeda
NICT
March 5, 2020

Hop-by-Hop Authentication in Content-Centric Networking/Named Data
Networking
draft-li-icnrg-hopauth-02

Abstract

The unpredictability of consumers, routers, copyholders, and publishers for the in-network data retrievals in Content-Centric Networking (CCN) / Named Data Networking (NDN) poses a challenge to design an authentication mechanism to inhibit the malicious consumers to flood data requests and prevent the fake data from being provided. Signature is adopted as the fundamental function in CCN / NDN, which however can only provide publisher authentication with additional certificate acquisition. This document describes the Hop-by-Hop Authentication mechanism (HopAuth) integrating certificate collection and packet forwarding potentially with the assistance from certificate authority to provide consumer authentication, copyholder authentication and path authentication to enable the in-network data retrieval to be trustworthy, besides the publisher authentication.

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 <https://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 September 6, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	5
3. System Descriptions	6
4. HopAuth Designs	7
4.1. Initial Trust Establishment	7
4.2. Data-centric Certificate Management	9
4.2.1. Certificate Exchange	9
4.2.2. Certificate Update and Revocation	9
4.3. Forwarding-Integrated Authenticable Data Retrieval	10
4.4. Suspension-Chain Model (SCM)	12
5. Protocol Message Format	12
6. Security Considerations	14
7. References	14
7.1. Normative References	14
7.2. Informative References	14
Authors' Addresses	16

1. Introduction

Information-Centric Networks in general, and Content-Centric Networking (CCN) [4] or Named Data Networking (NDN) [5] in particular, are the emerging network architectures enabling in-network caching and data retrievals through their names. In CCN/NDN, data can be cached at the intermediate routers, close to consumers for reducing delay and redundant bandwidth consumption or for the robustness under dynamic network environment. It has been noticed that CCN/NDN is a promising approach for the application scenarios in disaster networking [6], video streaming [7], and Internet of Things (IoT) [9].

In CCN/NDN, the basic network operations and these use scenarios with in-network data caching and retrievals lead the network to be seriously vulnerable under a variety of attacks, such as the impersonation attack, malicious-request attack [10], [11], [12], and the data poisoning attack [13], [14], [15]. The unpredictability of consumers, routers, copy holders, and publishers during data

retrievals in CCN/NDN poses the novel challenge to design data-centric authentication to prevent these attacks. This novel authentication should potentially enable the consumer authentication, copyholder authentication to authenticate the identity of the entity providing data, path authentication to authenticate the path from which data are retrieved, in addition to the publisher authentication.

On the other hand, signature is already adopted as the fundamental function in CCN/NDN, which promises to achieve the integrity and publisher authentication. It can partially prevent the above attacks and but still is insufficient to protect the unpredictable data retrievals in CCN/NDN. The unpredictability with which copy holders provide data, routers cache data, and consumers request data leads to great difficulty in inhibiting malicious-request attacks and data-poisoning attacks. To prevent data poisoning, consumers and routers need to verify data before caching them. If the data are found to be fake, the copy holder providing the data and the path to retrieve the data also need to be discovered in order to disable the further spread of that fake data. To prevent malicious-request attacks, copy holders need to verify the identities of the consumers.

There are many authentication mechanisms with key management schemes in the Internet, such as Kerberos [16], MSEC [17], X.509 [18], PGP [19], RPKI [20]. They are designed to achieve different purposes with centralized or decentralized approach based on end-to-end communication paradigm in the Internet. They can only provide the authentications of consumers and publishers or the link between them. In other words, they do not consider data-centric authentication, that is aimed to protect data in the networks.

For data-centric authentication, the key is to mitigate data-poisoning and malicious-request attacks. Regarding the data poisoning attacks, if the fake or corrupted data are cached along the path, consumers who use the path always retrieve the wrong data, because the router does not detect the cached data validity (as it is signed by attacker correctly). The fake or corrupted data are further cached, which pollute the routers as virus spreads. These routers will provide the fake or corrupted data to other potential consumers.

To mitigate the content poisoning attacks, the KeyId restriction has already been built in the CCNx design [3]. Only the requested data packet satisfying the interest with the KeyId restriction would be forwarded and stored in the CS, thus resulting in a reduction in the chances of cache poisoning. However, this approach requires users to obtain and keep the public key through certification authority (CA) for the potential content to be retrieved every time, which leads the

serious scalability problem and is also not suitable for the environment without capability of accessing the centralized servers to assert the authenticity of the key. It is also difficult to mitigate the content poisoning attacks mounted from the attacker controlling several or many intermediate routers or consumer nodes, because these nodes can collude with each other for data poisoning.

For malicious requests, even malicious Interests can be reduced by restricting sending rate, part of malicious Interests still can reach the copyholder. If copyholders verify the Interests before replying the data, the effect to inhibit the Interest flooding attack can be improved. For Interest verification, the first-hop router to consumer can remove the certificate from it to consumer before forwarding the Interest to preserve the consumer anonymity (See Section 4.3). Hence, the authentication from any consumer or router to Interest, data, copyholder, and the data retrieval path plays crucial role to inhibit the data poisoning and malicious-request attacks, besides the identity authentication to the publisher.

Another concern is that most of the existing authentication mechanisms rely on centralized servers (e.g. CA) to acquire keys or certificates, thereby increasing authentication delays, which we refer to herein as the delay-enlargement problem. Whenever data are transmitted and cached, router needs to contact centralized servers to retrieve certificates for authentications everytime. Obviously, these mechanisms cannot satisfy the following performance requirements for the emerging data-centric communication paradigm in CCN/NDN: any consumer or router needs to authenticate Interest, data, copyholder (including publisher), and the data retrieval path without additional messaging delay.

In this document, we propose a certificate chain-based approach named HopAuth, which enables data verification before caching and the authentications to Interest, data retrieval path, and copyholder for preventing the data poisoning attack and malicious-request attack. With HopAuth, forwarding-integrated hop-by-hop certificate collection is performed together with the adaptive replacement for parts of chain with highly trustworthy certificates, which can be operated independent of centralized servers. This specification derives directly from the design published in [21]. In HopAuth, a suspension-chain model (SCM) is used as the trust model, where the neighbor-trust-based certificate chain is suspended by CA-based trust. The HopAuth avoids reliance on centralized server(s) for chain construction and solves the delay-enlargement problem.

2. Terminology

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 [1].

The following terminology is used throughout this document.

- o Cryptographic key: A string of bits used by a cryptographic algorithm to transform plain-text into cipher-text or vice versa.
- o Signature: A cryptographic value calculated through public key algorithm from the data and a secret key only known by the signer. It is to validate the authenticity and integrity of a message.
- o Certificate: A data structure used to verifiably bind an identity to a cryptographic key.
- o Consumer: A node requesting data. It initiates communication by sending an interest packets.
- o Publisher: A node providing data. It originally creates or owns the data.
- o Router: A node forwarding data. It may hold memory to cache the data.
- o Forwarding Information Base (FIB): A lookup table in a router containing the name prefix and corresponding destination interface to forward the interest packets.
- o Pending Interest Table (PIT): A lookup table populated by the interest packets containing the name prefix of the requested data, and the outgoing interface used to forward the received data packets.
- o Content Store (CS): A storage space for a router to cache data objects. It is also known as in-network cache.
- o Physical entity: an entity that communicates using a physical device. This could be a router or a publisher node (PN) that hosts applications.
- o Logical entity: an entity that is involved in an application. This can be an authorizer, a sub-authorizer, or a publisher.

3. System Descriptions

Here we briefly explain the background and basic network operations of CCN/NDN. Different from the end-to-end communications in the Internet, CCN/NDN provides data-name-based retrievals as in Fig. 1. It further requires the data-centric authentication, instead of the current end-to-end secure channel establishment.

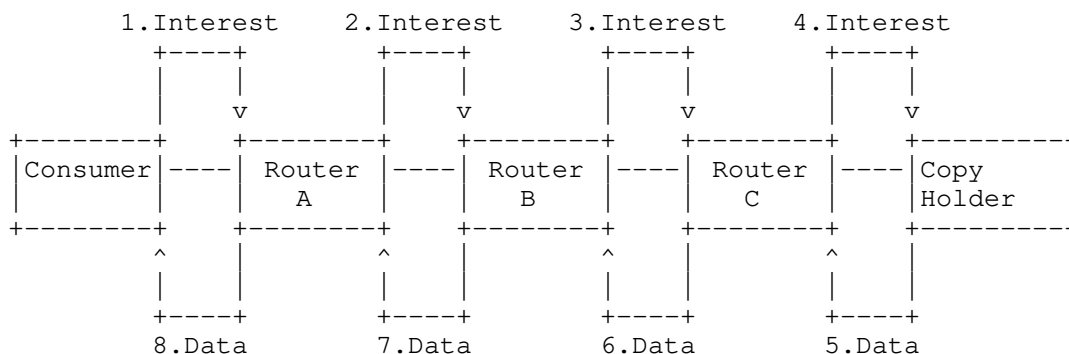


Figure 1: Request and reply messages forwarded by consumer, copy holder and routers.

In a CCN/NDN network, each router in a CCN/NDN network has three main data structures: a FIB for forwarding Interests, a CS for caching data, and a PIT for forwarding data. Basically there are two types of packets: interest and data. As in Fig. 1, consumer requests data by throwing an "interest" packet with the name of data to the network. Regarding the difference to note here between CCN [4] and NDN [5] is that in later versions of CCN, interest packet must carry a full data name, while in NDN it may carry a data name prefix.

Once a router receives an "interest" packet, it performs a series of the following look-up.

The router first checks in the CS to see whether it holds the corresponding data or not. If there is, it returns the data through the reverse path for forwarding interest packet as the copy holder in Fig. 1. If not, it performs a look-up of the PIT. If there is already a PIT entry matching the name of requested data, it is updated with the incoming interface of this new request and the interest is discarded. If there is no matching entry, it creates an entry in the PIT that lists the data name and the interfaces from which it received the interest. Then, the interest undergoes a FIB lookup to discover the outgoing interface.

Once a copy of the "data" packet is retrieved, the router sends it back to the data requester(s) using the trail of PIT entries and remove the PIT state every time that an interest is satisfied. Additionally, it may store the data in its CS.

However, data retrieval with in-network caching in CCN/NDN has been identified to suffer from malicious data-request attacks [10], [11], [12], and the data poisoning attacks [13], [14], [15]. In the former, adversaries impersonate consumers to create a flood of interests, and in the latter, they impersonate copy holders (e.g., routers or publishers) to provide fake data. These attacks are severe, because data are cached in a distributed manner, and copy holders have no way to verify consumers' identities, consumers/routers have no way to verify copy holders' identities to avoid caching fake data, and the path to retrieve data cannot be verified. This form of attack can quickly pollute the router caches as the virus spreads, because routers cache the fake data, redistribute them, and other intermediate routers re-cache them. It finally consumes much in-network caches and prevents consumers from retrieving the correct data.

4. HopAuth Designs

Herein we elaborate the basic design of HopAuth, which has three phases: 1) initial trust-establishment, 2) data-centric certificate management, and 3) forwarding-integrated authenticable data-retrieval. The trust model for HopAuth is given by a suspension chain model (SCM) later explained.

For the first phase, certificates are issued for neighboring entities and the certificate authority (CA) potentially issues certificates to a set of routers to form the highly trusted router group (HTRG). Let $CE(A \rightarrow B)$ represent the public-key certificate issued from A to B. In the second phase, routers exchange certificates within their neighborhoods, and any compromised entity can be shut down quickly. Finally, the third phase provides a hop-by-hop method for constructing a chain consisting of a physical-entity certificate chain (peCEChain) and a logical-entity certificate chain (leCEChain) for data-centric authentication.

4.1. Initial Trust Establishment

The initial trust establishment has two components: self-certifiable naming and certificate issuing.

Self-certifiable naming defines the rule for naming the principals, including entities, keys, and certificates, to enable the entities to be self-certifiable. As the extension of the cryptographically

generated address (CGA) [8], we merge the hash-based self-certifying names with hierarchical naming. It embeds the public key into the name, which is self-certifiable. The purpose of self-certifiable naming is to prevent stealing and spoofing of existing names. The public key of the name owner is bound cryptographically to the name. The name owner can use the corresponding private key to assert its ownership and to sign messages sent from the entity with that name. In fact, an attacker can create a new name from an arbitrary public key. However, the attacker cannot impersonate somebody else's name.

To issue a certificate, an entity should be convinced that a given public key truly belongs to another entity. Under that situation, it issues a certificate for this entity such that the public key is bound to the entity name by its signature. In HopAuth, certificate issuing is the initial trust establishment among entities having trust relationships, namely as neighbor-based trust for physical entities, CA-based trust, and authorization relationships for logical entities.

For neighbor-based trust, a physical entity creates a public key and the corresponding private key locally by itself. It generates the names using the public key based on the self-certifiable naming method. If physical entity B is a neighbor of entity A, B can announce its public key to A with the key name and A generates the certificate for B.

CA-based trust between two entities is established using the CA as the "introducer". The CA is managed by the network operator, and provides certificates to the owner's entities and the highly trusted physical entities close to them in the HTRG. The entities in the HTRG then confer CA-based trust relationships and issue certificates to each other.

All the highly trustable entities register at CA. CA has the whole view of these entities and determines the neighboring relations among the entities in HTRG. After the determination of the neighboring relations, an entity, A, sends interest to CA to request the certificates of its neighbors. CA replies with the related certificates to A, such as $CE(CA \rightarrow B)$. After A receiving the certificates, A verifies them, issues certificates from itself to those neighbors, such as $CE(A \rightarrow B)$, and sends to CA. CA verifies this certificate and forwards it to B when B requests.

In HopAuth, the physical entities pre-keep the certificates and associate certificates with interfaces in FIB. The physical entity knows the next hop of one interface, and it associates the certificate from that entity to itself with the forwarding interface.

This mechanism enables the appending of the relevant certificate to the packets as required when forwarding them.

For logical entities, the trust relationship is defined by the application. Each logical entity also generates a public/private key pair by itself. If logical entity A authorizes the right for entity B to manage a sub-category or publish data, A should provide a certificate for the true public key of B.

4.2. Data-centric Certificate Management

4.2.1. Certificate Exchange

Certificate exchange allows entities to share the certificates that they issue and hold. Each physical entity has a local repository in which to store certificates securely. In HopAuth, the physical entities request and keep all the certificates issued for or by their nearby highly trusted entities in the HTRG and by common neighboring physical entities. To exchange the information of certificates they hold, each entity hashes the names of certificates and exchange hashes to see if there is one missed in the neighborhood. If there is, the entity will send interest to its neighbor to retrieve that certificate. Finally, the physical entities hold all the certificates within a two-hop distance and the certificates with nearby highly trusted entities in the HTRG. These certificates are used to construct chains hop by hop, shorten certificate chains, and check the certificates appended by an up-stream entity. This certificate check is performed by the next hop of the entity to check whether the certificate appended by this entity is the same as the one stored by it. If the certificates are the same, the certificate passes the check. Otherwise, this packet will be dropped because of the fake certificate.

The certificates of the logical entities in an application should be stored in the repository of the PN. When data are published, the trust chain from the PN to the publisher can be automatically formed and appended to the packet. Meanwhile, the neighbors of the PN also keep all the certificates of the PN in order to check them during transmissions.

4.2.2. Certificate Update and Revocation

To guarantee its validity, each certificate in the network is issued with a certificate expiration time, after which the certificate is invalid.

For certificate updates using neighbor-based trust, the subject entity of the certificate should notify the issuer of its interest in

updating the certificate. On receiving this interest, the issuer checks whether this entity has been compromised. It then checks whether the mapping between the name and the public key satisfies the naming rule. If all the checks are passed, the issuer considers whether the public key of the subject entity is still trustworthy, generates an updated certificate, and replies with this updated certificate. If any check is failed, the issuer does not provide a certificate update to that entity. For certificate updates using CA-based trust in the HTRG, the subject entity of the certificate requests the CA to issue an updated certificate and provide it to the related nearby highly trusted routers, who can further issue update certificates.

If one entity no longer wishes to trust another entity, the former may revoke the certificate that it originally issued. Because certificates are issued over a one-hop distance, it is easy to detect the misbehavior of an entity. To revoke a certificate quickly, the revocation is announced over a two-hop distance. The revocation initiator broadcasts the revocation information to all the entities within a two-hop distance. Each entity receiving this information adds the compromised entity or certificate to its blacklist. All the packets from the compromised entities are dropped for a rapid local shutdown.

4.3. Forwarding-Integrated Authenticable Data Retrieval

Here, when forwarding interests and data, we define a forwarding-integrated hop-by-hop approach to construct the suspended chain from unpredictable copy holders to a consumer for authenticating interest, and from a consumer or router to data for authenticating copy holders or publishers or path to retrieve data. We let highly trusted routers replace the parts of chain with highly trusted certificates induced by CA's suspended trust to enhance trustworthiness.

The steps for data retrieval are as follows.

Step 1: The consumer issues an interest appended with its signature. It knows its next hop is the router to which it is connected. It then appends to this interest the certificate from the next-hop router to itself. Finally, it sends out the interest.

Step 2: When the interest is received by an router, the following operations will be performed. If the router is the first-hop router to the consumer, it verifies the correctness of the certificate from itself to the consumer and then remove this certificate before forwarding Interest to preserve the anonymity of consumer if the verification passes. Otherwise, it checks whether the previous certificates are correct. If the check succeeds and it belongs to

the HTRG, this router attempts to find the previous highly trusted router to replace the related part of the certificate chain with one highly trusted certificate in the suspended chain. If this router does not belong to the HTRG, it directly finds the interface to the next hop and appends to the interest the relevant certificate from the next-hop router to itself.

Step 3: This is executed if an router holds the requested data in its cache. The router checks the suspended chain from it to the consumer through the process described later in the SCM. If the verification succeeds, this router replies with the data packet. In the data packet, the suspended chain from this router to the publisher and the certificate from the next router to this router are appended. This router sends this data packet to the interface in reply as specified in the PIT.

Step 4: If the PN receives the interest, it verifies the suspended chain from itself to the consumer. If the verification succeeds, the PN discovers the suspended chain from itself to the publisher in its storage, and appends this certificate chain with the certificate from the first router to itself. Next, the PN replies with these data using the reverse path of the interest.

Step 5: After the router receives the data packet, it performs forwarding and possibly caching. When the router intends to cache the data, it first caches them in a temporary cache, which is separated from the data cache. Second, it checks the suspended chain from itself to the publisher. Only if the verification passes, these data can be cached in the data memory and the suspended chain from this router to the publisher will be cached along with the data. The verification is performed offline, which does not affect the speed of data retrieval. At the same time, this router checks the previous certificate. If the check is successful and it belongs to the HTRG, it will discover the previous entity in the suspended chain belonging to the HTRG. If there is a previous entity, the router replaces part of the related certificate path with a highly trusted certificate. Otherwise, the router directly finds the interface to the corresponding certificate from the next hop to itself and appends this certificate to the packet, then forwards this packet to the interface.

Step 6: After the consumer receives the data packet, there should be a certificate chain from it to the publisher. It verifies this suspended chain. If the verification passes, it believes that it gets the authentic public key of the publisher, and utilizes the key to verify the signature of the data. The consumer can also verify the copy holder or the routers on the path.

The constructed suspended chain can be further shortened by the routers to reduce the cost incurred by the suspended chain construction and verification. That is, routers can replace the verified suspended chain with a simplified certificate by specifying that this certificate is a shortened one.

The future network will consist of both the CCN/NDN routers and the routers without CCN/NDN functions. For such kind of partially deployed CCN/NDN network, the CA-based trust is used to logically connect the CCN/NDN routers to conceal the part of the network without CCN/NDN functions. That is, in Steps 2 and 5, the router attempts to find the previous highly trusted router to use one highly trusted certificate to link to the suspended certificate.

HopAuth can be further enhanced by the Blockchain with the function of the data life-cycle management for CCN/NDN [22].

4.4. Suspension-Chain Model (SCM)

A suspension-chain model (SCM) is the trust model in HopAuth. It is a flexible series of neighbor-trust-based certificates suspended by CA's trust, which form a suspension chain. In SCM, neighbor-based trust forms the certificate chain to realize data-centric authentication, whereas CA-based trust reduces the length of the chain to solve the trust degradation problem for certificate chain. For CA's trust, the CA assigns certificates to highly trusted routers as the suspension points based on CA's trust, which is the pre-trust between these routers and CA.

Public-key verification is the process for one entity to verify the authenticity of the public key of another entity. Entity X authenticates the public key of the entity Y by verifying the suspension chain in the following steps. First, X verifies the first certificate by its private key. If it is correct, each intermediate public key is used to verify the next direct associated certificate. This process continues for multiple rounds until the final certificate is verified. Finally, X obtains the true public key of the entity Y.

5. Protocol Message Format

HopAuth messages are encoded in the CCNx TLV format [2]. An example of HopAuth packet format with a certificate chain from consumer C0 to router R_n, (CE1(C0→R1), ..., CE_n(R_(n-1)→R_n)), is provided in Figure 2. As in Figure 2, the HopAuth message consists HopAuth header, and HopAuth certificate TLVs.

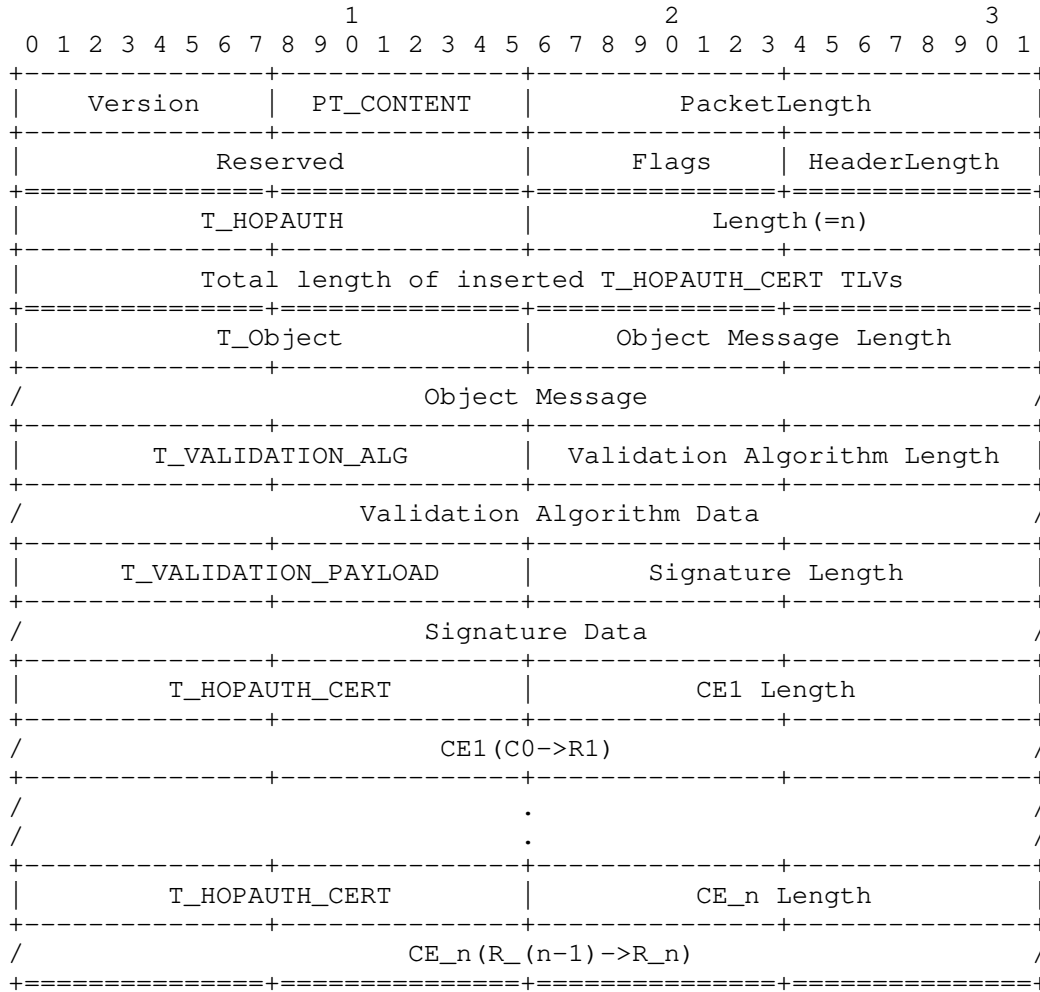


Figure 2: An Example of HopAuth packet format

The HopAuth header is composed of the fields of type, Length, and Total length of inserted T_HOPAUTH_CERT's TLVs. The type is T_HOPAUTH, the Length shows the number of inserted certificates, and Total length of inserted T_HOPAUTH_CERT's TLVs describes the total number of bits occupied by certificate TLVs. In the part of HopAuth certificate TLVs, the certificate name, length and certificate data are included. Take CE1(C0->R1) in Figure 2 as example. The type is T_HOPAUTH_CERT, CE1 Length is the length of certificate CE1(C0->R1), and CE1(C0->R1) is the certificate data for CE1(C0->R1).

6. Security Considerations

This document is entirely about an authentication mechanism in CCN/NDN.

7. References

7.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [2] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", RFC 8609, July 2019.

7.2. Informative References

- [3] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, July 2019.
- [4] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", Proc. CoNEXT, ACM, December 2009.
- [5] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and B. Zhang, "Named data networking", ACM Comput. Commun. Rev., vol. 44, no. 3, July 2014.
- [6] Seedorf, J., Arumaithurai, M., Tgami, A., Ramakrishnan, K., and N. Melazzi, "Research Directions for Using ICN in Disaster Scenarios", draft-irtf-icnrg-disaster-07 (work in progress), June 2019.
- [7] Westphal, C., Lederer, S., Posch, D., Timmerer, C., Azgin, A., Liu, W., Mueller, C., Detti, A., Corujo, D., Wang, J., Montpetit, M., and N. Murray, "Adaptive Video Streaming over Information-Centric Networking (ICN)", RFC 7933, August 2016.
- [8] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

- [9] Ravindran, R., Zhang, Y., Grieco, L., Lindgren, A., Burke, J., Ahlgren, B., and A. Azgin, "Design Considerations for Applying ICN to IoT", draft-irtf-icnrg-icniot-03 (work in progress), May 2019.
- [10] Afanasyev, A., Mahadevan, P., Moiseenko, I., Uzun, E., and L. Zhang, "Interest flooding attack and countermeasures in named data networking", Proc. IFIP Networking, IFIP, May 2013.
- [11] Compagno, A., Conti, M., Gasti, P., and G. Tsudik, "Poseidon: mitigating interest flooding ddos attacks in named data networking", Proc. LCN 2013, IEEE, October 2013.
- [12] Nguyen, T., Cogranne, R., and G. Doyen, "An optimal statistical test for robust detection against interest flooding attacks in ccn", Proc. International Symposium on Integrated Network Management (INM), IFIP/IEEE, May 2015.
- [13] Ghali, C., Tsudik, G., and E. Uzun, "Network-layer trust in named-data networking", ACM SIGCOMM Computer Communication Review, vol.44, no. 5, October 2014.
- [14] Kim, D., Nam, S., Bi, J., and I. Yeom, "Efficient content verification in named data networking", Proc. ACM Conference on Information-Centric Networking, ACM, September 2015.
- [15] Gasti, P., Tsudik, G., Uzun, E., and L. Zhang, "Dos and ddos in named data networking", Proc. IEEE ICCCN 2013, IEEE, August 2013.
- [16] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.
- [17] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.
- [18] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [19] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

- [20] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol Version 1", RFC 8210, September 2017.
- [21] Li, R., Asaeda, H., and J. Wu, "DCAuth: Data-Centric Authentication for Secure In-Network Big-Data Retrieval", IEEE Transactions on Network Science and Engineering, DOI: 10.1109/TNSE.2018.2872049, September 2018.
- [22] Li, R. and H. Asaeda, "A Blockchain-based Data Lifecycle Protection Framework for Information-Centric Network", IEEE Communications Magazine, vol. 57, no. 6, June 2019.

Authors' Addresses

Ruidong Li
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: lrd@nict.go.jp

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

ICNRG
Internet-Draft
Intended status: Experimental
Expires: 6 November 2022

D. Oran
Network Systems Research and Design
5 May 2022

Maintaining CCNx or NDN flow balance with highly variable data object
sizes
draft-oran-icnrg-flowbalance-07

Abstract

Deeply embedded in some ICN architectures, especially Named Data Networking (NDN) and Content-Centric Networking (CCNx) is the notion of flow balance. This captures the idea that there is a one-to-one correspondence between requests for data, carried in Interest messages, and the responses with the requested data object, carried in Data messages. This has a number of highly beneficial properties for flow and congestion control in networks, as well as some desirable security properties. For example, neither legitimate users nor attackers are able to inject large amounts of un-requested data into the network.

Existing congestion control approaches however have a difficult time dealing effectively with a widely varying MTU of ICN data messages, because the protocols allow a dynamic range of 1-64K bytes. Since Interest messages are used to allocate the reverse link bandwidth for returning Data, there is large uncertainty in how to allocate that bandwidth. Unfortunately, most current congestion control schemes in CCNx and NDN only count Interest messages and have no idea how much data is involved that could congest the inverse link. This document proposes a method to maintain flow balance by accommodating the wide dynamic range in Data message size.

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 <https://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 6 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	2
2. Requirements Language	4
3. Method to enhance congestion control with signaled size information in Interest Messages	4
3.1. How to predict the size of returning Data messages	5
3.2. Handling 'too big' cases	6
3.3. Handling 'too small' cases	7
3.4. Interactions with Interest Aggregation	8
3.5. Operation when some Interests lack the expected data size option and some have it	10
4. Dealing with malicious actors	11
5. Mapping to CCNx and NDN packet encodings	12
5.1. Packet encoding for CCNx	12
5.2. Packet encoding for NDN	12
6. IANA Considerations	13
7. Security Considerations	13
8. Acknowledgements	13
9. References	13
9.1. Normative References	13
9.2. Informative References	13
Author's Address	15

1. Introduction

Deeply embedded in some ICN architectures, especially Named Data Networking ([NDN]) and Content-Centric Networking (CCNx [RFC8569],[RFC8609]) is the notion of flow balance. This captures the idea that there is a one-to-one correspondence between requests for data, carried in Interest messages, and the responses with the requested data object, carried in Data messages. This has a number of highly beneficial properties for flow and congestion control in networks, as well as some desirable security properties. For example, neither legitimate users nor attackers are able to inject

large amounts of un-requested data into the network.

This approach leads to a desire to make the size of the objects carried in Data messages small and near constant, because flow balance can then be kept using simple bookkeeping of how many Interest messages are outstanding. While simple, constraining Data messages to be quite small - usually on the order of a link Maximum Transmission Unit (MTU) - has some constraints and deleterious effects, among which are:

- * Such small data objects are inconvenient for many applications; their natural data object sizes can be considerably larger than a link MTU.
- * Applications with truly small data objects (e.g. voice packets in an Internet telephony applications) have no way to communicate that to the network, causing resources to still be allocated for MTU-sized data objects
- * When chunking a larger data object into multiple Data messages, each message has to be individually cryptographically hashed and signed, increasing both computational overhead and overall message header size. The signature can be elided when Manifests are used (by signing the Manifest instead), but the overhead of hashing multiple small messages rather than fewer larger ones remains.

One approach which helps with the last of these is to employ fragmentation for Data messages larger than the Path MTU (PMTU). Such messages are carved into smaller pieces for transmission over the link(s). There are three flavors of fragmentation: end-to-end, hop-by-hop with reassembly at every hop, and hop-by-hop with cut-through of individual fragments. A number of ICN protocol architectures incorporate fragmentation and schemes have been proposed for both NDN and CCNx, for example in [Ghali2013]. Fragmentation alone does not ameliorate the flow balance problem however, since from a resource allocation standpoint both memory and link bandwidth must be set aside for maximum-sized data objects to avoid congestion collapse under overload.

The design space considered in this document does not however extend to arbitrarily large objects (e.g. 100's of kilobytes or larger). As the dynamic range of data object sizes gets very large, finding the right tradeoff between handling a large number of small data objects versus a single very large data object when allocating link and buffer resources becomes intractable. Further, the semantics of Interest-Data exchanges means that any error in the exchange results in a re-issue of an Interest for the entire Data object. Very large data objects represent a performance problem because the cost of

retransmission when Interests are retransmitted (or re-issued) becomes unsustainably high. Therefore, the method we propose deals with a dynamic range of object sizes from very small (a fraction of a link MTU) to moderately large - about 64 kilobytes or equivalently about 40 Ethernet packets, and assumes an associated fragmentation scheme to handle link MTUs that cannot carry the Data message in a single link-layer packet.

The approach described in the rest of this document maintains flow balance under the conditions outlined above by allocating resources accurately based on expected Data message size, rather than employing simple interest counting.

2. 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].

3. Method to enhance congestion control with signaled size information in Interest Messages

Before diving into the specifics of the design, it is useful to consider how congestion control works in NDN/CCNx. Unlike the IP protocol family, which relies on end-to-end congestion control (e.g. TCP[RFC0793], DCCP[RFC4340], SCTP[RFC4960], QUIC[I-D.ietf-quic-transport]), CCNx and NDN employ hop-by-hop congestion control. There is per-Interest/Data state at every hop of the path and therefore for each outstanding Interest, bandwidth for data returning on the inverse path can be allocated. In many current designs, this allocation is done using simple Interest counting - by queueing and subsequently forwarding one Interest message from a downstream node, implicitly this provides a guarantee (either hard or soft) that there is sufficient bandwidth on the inverse direction of the link to send back one Data message. A number of congestion control schemes have been developed that operate in this fashion, for example [Wang2013], [Mahdian2016], [Song2018], [Carofiglio2012]. Other schemes, like [Schneider2016] neither count nor police interests, but instead monitor queues using AQM (active queue management) to mark or drop returning Data packets that have experienced congestion. It is worth noting that every congestion control algorithm has an explicit fairness goal and associated objective function (usually either [minmaxfairness] or [proportionalfairness]). If your fairness is to be based on resource usage, pure interest counting doesn't do the trick, since a consumer asking for large thing can saturate a link and shift loss to consumers asking for small things.

In order to deal with a larger dynamic range of Data message size, some means is required to allocate link bandwidth for Data messages in bytes with an upper bound larger than a Path MTU and a lower bound lower than a single link MTU. Since resources are allocated for returning Data based on arriving Interests, this information must be available in Interest messages.

Therefore, one key idea is the inclusion of an `_expected data size_` TLV in each Interest message. This allows each forwarder on the path taken by the Interest to more accurately allocate bandwidth on the inverse path for the returning Data message. Also, by including the expected data size, large objects will have a corresponding weight in resource allocation, maintaining link and forwarder buffering fairness. The simpler Interest counting scheme was nominally "fair" on a per-exchange basis within the variations of data that fit in a single PMTU packet because all Interests produced similar amounts of data in return. In the absence of such a field, it is not feasible to allow a large dynamic range in object size. While schemes like [Schneider2016] would not employ the expected data size to allocate reverse link bandwidth, they can still benefit from the information to affect the AQM congestion marking algorithm, preferentially marking data packets that exceed the expected data size from the corresponding Interest.

It is natural to ask whether the additional complexity introduced into an ICN forwarder, and the additional computational cost for the congestion control operations is worthwhile. For congestion control schemes like [Schneider2016] the additional overhead is not trivial, since no Interest counting is happening. However, if a congestion control is `_already_` counting Interests, the additional overhead is minimal, only reading one extra TLV from the Interest and incrementing the outstanding data amount for the corresponding queue by that number rather than a constant of 1. The overhead on returning data is simply reducing the amount by the actual Data message size, rather than by 1.

3.1. How to predict the size of returning Data messages

This of course raises the question "How does the requester know how big the corresponding Data message coming back will be?". For a number of important applications, the size is known a priori due to the characteristics of the application. Here are some examples:

- * For many sensor and other Internet-of-Things applications, the data is instrument readings which have fixed known size.

- * In video streaming, the data is output of a video encoder which produces variable sized frames. This information is typically made available ahead of time to the streaming clients in the form of a `_Manifest_` (e.g [DASH], FLIC [I-D.irtf-icnrg-flic]), which contains the names of the corresponding segments (or individual frames) of video and audio and their sizes.
- * Internet telephony applications use vocoders that typically employ fixed-size audio frames. Therefore, their size is known either a priori, or via an initialization exchange at the start of an audio session.

The more complex cases arise where the data size is not known at the time the Interest must be sent. Much of the nuance of the proposed scheme is in how mismatches between the expected data size and the actual Data message returned are handled. The consumer can either under- or over-estimate the data size. In the former case, the under-estimate can lead to congestion and possible loss of data. In the latter case, bandwidth that could have been used by data objects requested by other consumers might be wasted. We first consider "honest" mis-estimates due to imperfect knowledge by the ICN application; later we consider malicious applications that are using the machinery to mount some form of attack. We also consider the effects of Interest aggregation if the aggregated Interests have differing expected data sizes. Also, it should be obvious that if the Data message arrives, the application learns its actual size, which may or may not be useful in adjusting the expected data size estimate for future Interests.

In all cases, the expected data size from the Interest can be incorporated in the corresponding Pending Interest Table (PIT) entry of each CCNx/NDN forwarder on the path and hence when a (possibly fragmented) Data object comes back, its total size is known and can be compared to the expected size in the PIT for a mismatch. Aside: In the case of fragmentation, we assume a fragmentation scheme in which the total Data message size can be known as soon as any one fragment is received (a reasonable assumption for most any well-designed fragmentation method, such as that in [Ghali2013]).

3.2. Handling 'too big' cases

If the returning Data message is larger than the expected data size, the extra data could result in either unfair bandwidth allocation or possibly data loss under congestion conditions. When this is detected, the forwarder has three choices:

1. It could forward the Data message anyway, which is safe under non-congestion conditions, but unfair and possibly unstable when the output link is congested
2. It could forward the data when un-congested (e.g. by assessing output queue depth) but drop it when congested
3. It could always drop the data, as a way of "punishing" the requester for the mis-estimate.

Either of the latter two strategies is acceptable from a congestion control point of view. However, it is not a good idea to simply drop the Data message with no feedback to the issuer of the Interest because the application has no way to learn the actual data size and retry. Further, recovery would be delayed until the failing Interest timed out. Therefore, an additional element needed in protocol semantics is the incorporation of a "Data too big" error message (achieved via the use of an "Interest Return" packet in CCNx).

Upon dropping data as above, the CCNx/NDN forwarder converts the normal Data message into an Interest Return packet containing the existing [RFC8609] T_MTU_TOO_LARGE error code and the actual size of the Data message instead of the original content. It propagates that back toward the client identically to how the original Data message would have been handled. Subsequent nodes upon receiving the T_MTU_TOO_LARGE error treat identically to other Interest Return errors. When the Interest Return eventually arrives back to the issuer of the Interest, the user MAY reissue the Interest with the correct expected data size.

One detail to note is that an Interest Return carrying T_MTU_TOO_LARGE must be deterministically smaller than the expected data size in all cases. This is clearly the case for large data objects, but there is a corner case with small data objects. There has to be a minimum expected data size that a client can specify in their Interests, and that minimum cannot be smaller than the size of a T_MTU_TOO_LARGE Interest Return packet.

3.3. Handling 'too small' cases

Next we consider the case where the returning data is smaller than the expected data size. While this case does not result in congestion, it can cause resources to be inefficiently allocated because not all of the set-aside bandwidth for the returning data object gets used. The simplest and most straightforward way to deal with this case is to essentially ignore it. The motivation for not worrying about the smaller data mismatch is that in many situations that employ usage-based resource measurement (and possibly charging),

it is trivial to just account for the usage according to the larger expected data size rather than actual returned data size. Properly adjusting congestion control parameters to somehow penalize users for over-estimating their resource usage requires fairly heavyweight machinery, which in most cases is not warranted. If desired, any of the following mechanisms could be considered:

- * Attempt to identify future Interests for the same object or closely related objects and allocate resources based on some retained state about the actual size of prior objects
- * Police consumer behavior and decrease the expected data size in one or more future Interests to compensate
- * For small objects, do more optimistic resource allocation on the links on the presumption that there will be some "slack" due to clients overestimating data object size.

3.4. Interactions with Interest Aggregation

One protocol detail of CCNx/NDN that needs to be dealt with is Interest Aggregation. Interest Aggregation, while a powerful feature for maintaining flow balance when multiple consumers send Interests for the same Named object, introduces subtle complications. Whenever a second or subsequent Interest arrives at a forwarder with an active PIT entry it is possible that those Interests carry different parameters, for example hop limit, payload, etc. It is therefore necessary to specify the exact behavior of the forwarder for each of the parameters that might differ. In the case of the expected data size parameter defined here, the value is associated with the ingress face on which the Interest creating the PIT entry arrived, as opposed to being global to the PIT entry as a whole. Interest aggregation interacts with expected data size if Interests from different clients contain different values of the expected data size. As above in Section 3.3, the simplest solution to this problem is to ignore it, as most error cases are benign. However, there is one problematic error case where one client provides an accurate expected data size, but another who issued the Interest first underestimates, causing both to receive a T_MTU_TOO_LARGE error. This introduces a denial of service vulnerability, which we discuss below together with the other malicious actor cases.

There are two cases to consider:

1. The arriving Interest carries an expected data size smaller than any of the values associated with the PIT entry.

2. The arriving Interest carries an expected data size larger than any of the values associated with the PIT entry.

For Case (1) the Interest can be safely aggregated since the upstream links will have sufficient bandwidth allocated based on the larger expected data size (assuming the original Interest's expected data size was itself sufficiently large to accommodate the actual size of the returning Data). On the other hand, should the incoming face have bandwidth allocated based on the larger existing Interest's expected data size, or on the smaller value in the arriving interest? Here there are two possible approaches:

- a. Allocate based on the data size already in the PIT. In this case the consumer sending the earlier Interest can cause over-allocation of link bandwidth for other incoming faces, but there will not be a T_MTU_TOO_LARGE error generated for that Interest
- b. Allocate based on the value in the arriving Interest. If the returning Data is in fact larger, generate a T_MTU_TOO_LARGE Interest Return on that ingress face, while successfully returning the Data message on any faces that do not exhibit a too small expected data size

It is RECOMMENDED that the second policy be followed. The reasons behind this recommendation are as follows:

1. The link can be congested quite quickly after the queuing decision is made, especially if the data has a long link-occupancy time, so this is a safer alternative.
2. The cost of returning the error is only one link RTT, since the consumer (or downstream forwarder) can immediately re-issue the Interest with the correct size and perhaps pick up the cached object from the upstream forwarder's Content Store.
3. Being optimistic and returning the data interacts with the behavior of aggregate resource control and resource accounting, which in turn raises the messy issue of whether to "charge" the consumer for the actual bandwidth used or only for the requested bandwidth in the expected data.
4. The rabbit hole goes deeper if you add differential QoS to the equation or consumers "playing games" and intentionally underestimating so their interests get satisfied when links aren't congested. This makes handling malicious actors (Section 4) more difficult.

For Case (2) above, the Interest MUST be forwarded rather than aggregated to prevent a consumer from mounting a denial of service attack by sending intentionally too small expected data size (see Section 4 for additional detail on this and other attacks). As above for Case (1) it is RECOMMENDED that policy (b) above be followed.

3.5. Operation when some Interests lack the expected data size option and some have it

Since the expected data size is an optional hop-by-hop packet field, forwarders need to be prepared to handle an arbitrary mix of packets containing or lacking this option. There are two general things to address.

First, we assume that any forwarder supporting expected data size is running a more sophisticated congestion control algorithm than one employing simple interest counting. The link bandwidth resource allocation is therefore based directly, or indirectly, on the expected Data size in bytes. Therefore, the forwarder has to assign a value to use in the resource allocation for the reverse link. This specification does not mandate any particular approach or a default value to use. However, in the absence of other guidance, it makes sense to do one of two things:

1. Pick a default based on the link MTU of the face on which the Interest arrived and use that for all Interests lacking an expected data size. This is likely to be most compatible with simple interest counting which would rate limit all incoming interests equally.
2. Configure some values for given Name prefixes that have known sizes. This may be appropriate for dedicated forwarders supporting single use cases, such as:
 - * A forwarder handling IoT sensors sending very small Data messages
 - * A forwarder handling real-time video with large average Data packets that exceed link MTU and are routinely fragmented
 - * A forwarder doing voice trunking where the vocoders produce moderate sized packets, still much smaller than the link MTU

The second area to address is what to do if an interest lacking an expected Data size is responded to by a Data message whose size exceeds the default discussed above. It would be inappropriate to issue a T_MTU_TOO_LARGE error, since the consumer is unlikely to understand or deal correctly with that new error case. Instead, it is RECOMMENDED that the forwarder:

- * Ignore the mismatch if the reverse link is not congested and return the requested Data message anyway.
- * If the reverse link is congested, issue an Interest Return with the T_NO_RESOURCES error code

This specification does not define or recommend any particular algorithm for assessing the congestion state of the link(s) to carry the Data message downstream to the requesting consumers. It is assumed that a reasonable algorithm is in use, because otherwise even basic Interest counting forms of congestion control would not be effective.

4. Dealing with malicious actors

First we note that various known attacks in CCNx or NDN can also be mounted by users employing this method. Attacks that involve interest flooding, cache pollution, cache poisoning, etc. are neither worsened nor ameliorated by the introduction of the congestion control capabilities described here. However, there are two new vulnerabilities that need to be dealt with. These two new vulnerabilities involve intentional mis-estimation of data size.

The first is a consumer who intentionally over-estimates data size with the goal of preventing other users from using the bandwidth. This is at most a minor additional concern given the discussion of how to handle over-estimation by honest clients in Section 3.2. If one of the amelioration techniques described there is used, the case of malicious over-estimation is also dealt with adequately.

The second is a user who intentionally under-estimates the data size with the goal having its Interest processed while the other aggregated interests are not processed, thereby causing T_MTU_TOO_LARGE errors and denying service to the other users with overlapping requests. There are a number of possible mitigation techniques for this attack vector, ranging in complexity. We outline two below; there may be others as or more effective with acceptable complexity and overhead:

- * (Simplest) A user sending Interests resulting in a T_MTU_TOO_LARGE error is treated similarly to users mounting interest flooding attacks; the a router aggregating Interests with differing expected data sizes rate limits the face(s) exhibiting these errors, thus decreasing the ability of a user to issue enough mis-estimated Interests to collide and generate Interest aggregation.
- * An ICN forwarder aggregating Interests remembers in the PIT entry not only the expected data size of the Interest it forwarded, but the maximum of the expected data size of the other Interests it aggregated. If a T_MTU_TOO_LARGE error comes back, instead of propagating it, the forwarder MAY treat this as a transient error, drop the Interest Return, and re-forward the Interest using the maximum expected data size in the PIT (assuming it is bigger). This recovers from the error, but the attacker can still cause an extra round trip to the producer or to an upstream forwarder with a copy of the data in its Content Store.

5. Mapping to CCNx and NDN packet encodings

The only actual protocol needed is a TLV in Interest messages that states the size in bytes of the expected Data Message coming back, and in the Interest Return on a "too big" error to carry the actual data size. In the case of CCNx, this covers the encapsulated Data Object, but not the hop-by-hop headers.

5.1. Packet encoding for CCNx

For CCNx[RFC8569] there is a new hop-by-hop header TLV, and a new value of the Interest Return "Return Type".

Expected Data Size (for Interest messages), or Actual Data Size (for Interest Return messages) TLV

Abbrev	Name	Description
T_DATASIZE	Data Size	Expected (Section 3) or Actual (Section 3.2) Data Size

Table 1: Data Size TLV

5.2. Packet encoding for NDN

TBD based on [NDNTLV]. Suggestions from the NDN team greatly appreciated.

6. IANA Considerations

Please Add the T_DATASIZE TLV to the Hop-by-Hop TLV types registry of RFC8609, with fixed length of 2, and data type numeric

Expected/Actual Data Size TLV encoding. The range has an upper bound of 64K bytes, since that is the largest MTU supported by CCNx.

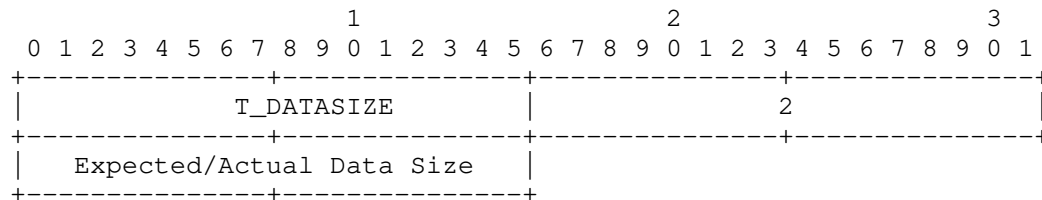


Figure 1: Expected/Actual Dataize using RFC8609 encoding

7. Security Considerations

Section 4 addresses the major security considerations for this specification.

8. Acknowledgements

Klaus Schneider and Ken Calvert have contributed a number of useful comments which have substantially improved the document.

9. References

9.1. 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.

9.2. Informative References

- [Carofiglio2012]
Carofiglio, G., Gallo, M., and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks, in ICN Workshop at SIGcomm 2012", DOI 10.1145/2377677.2377772, 2102,
<<http://conferences.sigcomm.org/sigcomm/2012/paper/icn/p37.pdf>>.
- [DASH] "Dynamic Adaptive Streaming over HTTP", various,
<https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP>.
- [Ghali2013]
Ghali, C., Narayanan, A., Oran, D., Tsudik, G., and C. Wood, "Secure Fragmentation for Content-Centric Networks, in IEEE 14th International Symposium on Network Computing and Applications", DOI 10.1109/nca.2015.34, 2015,
<<http://dx.doi.org/10.1109/NCA.2015.34>>.
- [I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-27, 21 February 2020,
<<https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27>>.
- [I-D.irtf-icnrg-flic]
Tschudin, C., Wood, C., Mosko, M., and D. Oran, "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-02, 4 November 2019,
<<https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-flic-02>>.
- [Mahdian2016]
Mahdian, M., Arianfar, S., Gibson, J., and D. Oran, "MIRCC: Multipath-aware ICN Rate-based Congestion Control, in Proceedings of the 3rd ACM Conference on Information-Centric Networking", DOI 10.1145/2984356.2984365, 2016,
<<http://conferences2.sigcomm.org/acm-icn/2016/proceedings/p1-mahdian.pdf>>.
- [minmaxfairness]
"Max-min Fairness", various,
<https://en.wikipedia.org/wiki/Max-min_fairness>.
- [NDN] "Named Data Networking", various,
<<https://named-data.net/project/execsummary/>>.

- [NDNTLV] "NDN Packet Format Specification.", 2016,
<<http://named-data.net/doc/ndn-tlv/>>.
- [proportionalfairness]
"Proportionally Fair", various,
<https://en.wikipedia.org/wiki/Proportionally_fair>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
RFC 793, DOI 10.17487/RFC0793, September 1981,
<<https://www.rfc-editor.org/info/rfc793>>.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
Congestion Control Protocol (DCCP)", RFC 4340,
DOI 10.17487/RFC4340, March 2006,
<<https://www.rfc-editor.org/info/rfc4340>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol",
RFC 4960, DOI 10.17487/RFC4960, September 2007,
<<https://www.rfc-editor.org/info/rfc4960>>.
- [Schneider2016]
Schneider, K., Yi, C., Zhang, B., and L. Zhang, "A
Practical Congestion Control Scheme for Named Data
Networking, in Proceedings of the 2016 conference on 3rd
ACM Conference on Information-Centric Networking - ACM-ICN
'16", DOI 10.1145/2984356.2984369, 2016,
<[http://conferences2.sigcomm.org/acm-icn/2016/proceedings/
p21-schneider.pdf](http://conferences2.sigcomm.org/acm-icn/2016/proceedings/p21-schneider.pdf)>.
- [Song2018] Song, J., Lee, M., and T. Kwon, "SMIC: Subflow-level
Multi-path Interest Control for Information Centric
Networking, in 5th ACM Conference on Information-Centric
Networking", DOI 10.1145/3267955.3267971, 2018,
<[https://conferences.sigcomm.org/acm-icn/2018/proceedings/
icn18-final62.pdf](https://conferences.sigcomm.org/acm-icn/2018/proceedings/icn18-final62.pdf)>.
- [Wang2013] Wang, Y., Rozhnova, N., Narayanan, A., Oran, D., and I.
Rhee, "An Improved Hop-by-hop Interest Shaper for
Congestion Control in Named Data Networking, in ACM
SIGCOMM Workshop on Information-Centric Networking",
DOI 10.1145/2534169.2491233, 2013,
<[http://conferences.sigcomm.org/sigcomm/2013/papers/icn/
p55.pdf](http://conferences.sigcomm.org/sigcomm/2013/papers/icn/p55.pdf)>.

Author's Address

Dave Oran
Network Systems Research and Design
4 Shady Hill Square
Cambridge, MA 02138
United States of America
Email: daveoran@orandom.net

ICNRG
Internet-Draft
Intended status: Experimental
Expires: 6 November 2022

I. Moiseenko
UCLA
D. Oran
Network Systems Research and Design
5 May 2022

Path Steering in CCNx and NDN
draft-oran-icnrg-pathsteering-06

Abstract

Path Steering is a mechanism to discover paths to the producers of ICN content objects and steer subsequent Interest messages along a previously discovered path. It has various uses, including the operation of state-of-the-art multipath congestion control algorithms and for network measurement and management. This specification derives directly from the design published in Path Switching in Content Centric and Named Data Networks (4th ACM Conference on Information-Centric Networking - ICN'17) and therefore does not recapitulate the design motivations, implementation details, or evaluation of the scheme. Some technical details are different however, and where there are differences, the design documented here is to be considered definitive.

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 <https://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 6 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	2
1.1. Path Steering as an experimental extension to ICN protocol architectures	3
1.2. Requirements Language	3
1.3. Terminology	4
2. Essential elements of ICN path discovery and path steering .	4
2.1. Path Discovery	4
2.2. Path Steering	6
2.3. Handling Path Steering errors	7
2.4. How to represent the Path Label	8
3. Mapping to CCNx and NDN packet encodings	9
3.1. Path label TLV	9
3.2. Path label encoding for CCNx	10
3.3. Path label encoding for NDN	11
4. IANA Considerations	12
5. Security Considerations	12
5.1. Cryptographic protection of a path label	14
6. References	15
6.1. Normative References	15
6.2. Informative References	16
Authors' Addresses	17

1. Introduction

Path Steering is a mechanism to discover paths to the producers of ICN content objects and steer subsequent Interest messages along a previously discovered path. It has various uses, including the operation of state-of-the-art multipath congestion control algorithms and for network measurement and management. This specification derives directly from the design published in [Moiseenko2017] and therefore does not recapitulate the design motivations, implementation details, or evaluation of the scheme. That publication should be considered a normative reference as it is not likely a reader will be able to understand all elements of this design without first having read the reference. Some technical details are different however, and where there are differences, the design documented here is to be considered definitive.

Path discovery and subsequent path steering in ICN networks is facilitated by the symmetry of forward and reverse paths in the CCNx and NDN architectures. Path discovery is achieved by a consumer endpoint transmitting an ordinary Interest message and receiving a Content (Data) message containing an end-to-end path label constructed on the reverse path by the forwarding plane. Path steering is achieved by a consumer endpoint including a path label in the Interest message, which is forwarded to each nexthop through the corresponding egress interfaces in conjunction with longest name prefix match (LNPM) FIB lookup.

1.1. Path Steering as an experimental extension to ICN protocol architectures

There are a number of important use cases to justify extending ICN architectures such as CCNx [RFC8569] or NDN [NDN] to provide these capabilities. These are summarized as follows:

- * Support the discovery, monitoring and troubleshooting of multi-path network connectivity based on names and name prefixes. Analogous functions have been shown to be a crucial operational capability in multicast and multi-path topologies for IP. The canonical tools are the well-known `_traceroute_` and `_ping_`. For point-to-multipoint MPLS the more recent tree trace [RFC8029] protocol is used. Equivalent diagnostic functions have been defined for CCNx through the ICN Ping [I-D.irtf-icnrg-icnping] and ICN Traceroute [I-D.irtf-icnrg-icntraceroute] specifications, both of which are capable of exploiting path steering if available.
- * Perform accurate online measurement of network performance, which generally requires multiple consecutive packets follow the same path under control of an application.
- * Improve the performance and flexibility of multi-path congestion control algorithms. Congestion control schemes such as [Mahdian2016] and [Song2018] depend on the ability of a consumer to explicitly steer packets onto individual paths in a multi-path and/or multi-destination topology.
- * A consumer endpoint can mitigate content poisoning attacks by directing its Interests onto the network paths that bypass poisoned caches.

1.2. 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].

1.3. Terminology

This document uses the general ICN terms that are defined in [RFC8793]. In addition we define the following terms specific to path steering:

Path Discovery: The process of sending an Interest requesting discover of a path and if successful, receiving a Data containing a Path Label for the path the corresponding Interest traversed

Path Steering: The process of sending an Interest message containing the Path Label of a previously discovered path in order that the forwarders use that path when forwarding that particular Interest message.

Path Label: An optional field in the packet indicating a particular path from a consumer to either a producer, or a forwarder cache that can respond with the requested item. In an Interest message, the Path Label gets built up hop by hop as the interest traverses a path. In a Data message, the Path Label carries the full path information back to the consumer for use in one or more subsequent Interest messages.

Nexthop Label: One entry in a Path Label representing the next hop for the corresponding forwarder to use when a path-steered Interest message arrives at that forwarder. A sequence of Nexthop Labels constitutes a full Path Label.

2. Essential elements of ICN path discovery and path steering

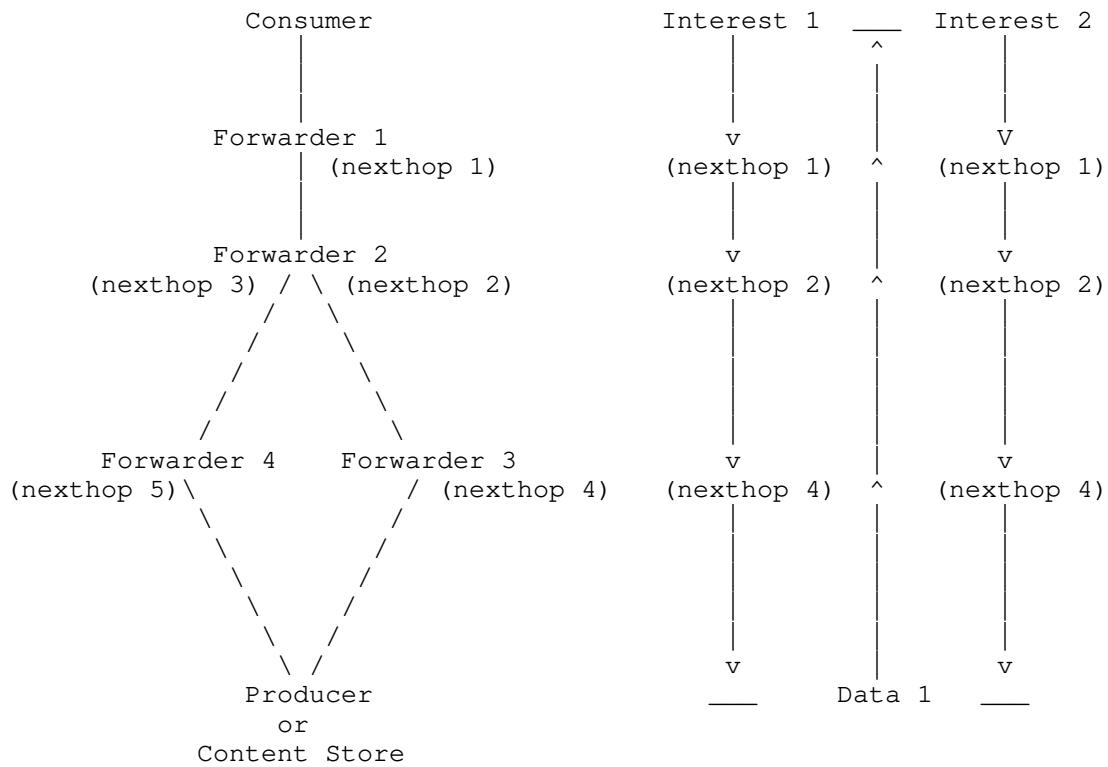
We elucidate the design using CCNx semantics [RFC8569] and extend its Packet Encoding [RFC8609] as defined in Section 3.2. While the terminology is slightly different, this design can be applied also to NDN, by extending its bespoke packet encodings [NDNTLV] (See Section 3.3).

2.1. Path Discovery

End-to-end Path Discovery for CCNx is achieved by creating a path label and placing it as a hop-by-hop TLV in a CCNx Content (Data) message. The path label is constructed hop-by-hop as the message traverses the reverse path of transit CCNx forwarders as shown in the first example in Figure 1. The path label is updated by adding to the existing path label the nexthop label of the interface at which the Content (Data) message has arrived. Eventually, when the Content(Data) message arrives at the consumer, the path label identifies the complete path the Content (Data) message took to reach the consumer. As shown in the second example in the figure, when

multiple paths are available, subsequent interests can discover additional paths by omitting a path steering TLV and obtaining a new path label on the returning interest.

Discover and use first path:



Discover and use second path:

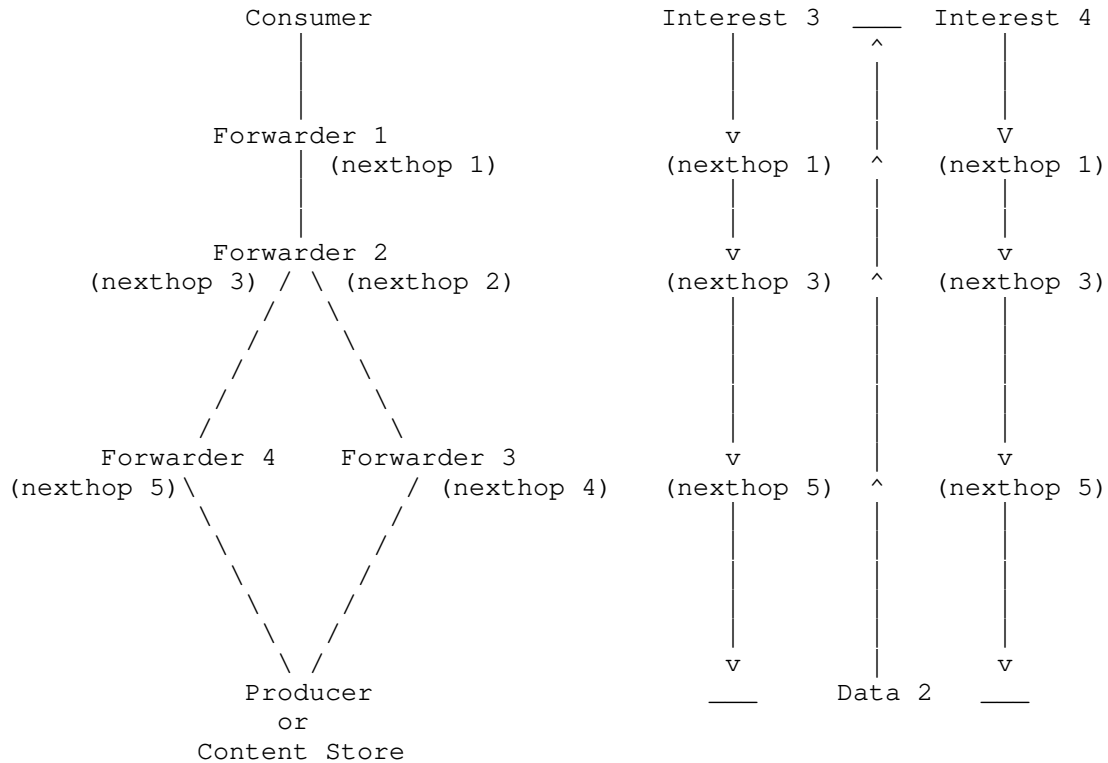


Figure 1: Basic example of path discovery and steering

2.2. Path Steering

Due to the symmetry of forward and reverse paths in CCNx, a consumer application can reuse a discovered path label to fetch the same or similar (e.g. next chunk, or next Application Data Unit, or next pointer in a Manifest [I-D.irtf-icnrg-flic]) Content (Data) message over the discovered network path. This Path Steering is achieved by processing the Interest message's path label at each transit ICN forwarder and forwarding the Interest through the specified nexthop among those identified as feasible by LNPM FIB lookup (Figure 2).

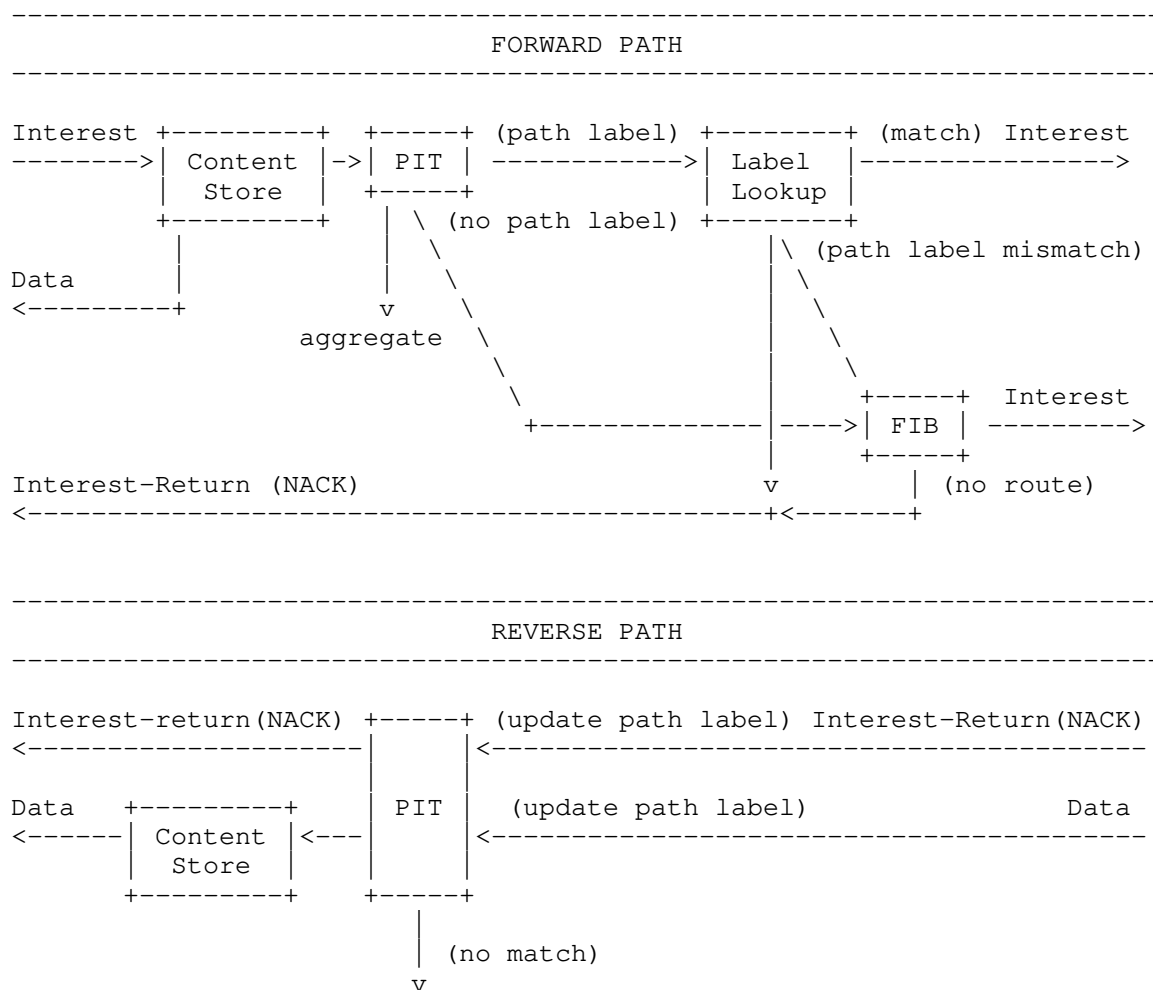


Figure 2: Path Steering CCN / NDN data plane

2.3. Handling Path Steering errors

Over time, the state of interfaces and the FIB on forwarders may change such that, at any particular forwarder, a given nexthop is no longer valid for a given prefix. In this case, the path label will point to a now-invalid nexthop. This is detected by failure to find a match between the decoded nexthop ID and the nexthops of the FIB entry after LNPM FIB lookup.

On detecting an invalid path label, the forwarder SHOULD respond to the Interest with an Interest-Return. We therefore define a new `_Invalid path label_` response code for the Interest Return message and include the current path label as a hop-by-hop header. Each transit forwarder processing the Interest-Return message updates the path label in the same manner as Content (Data) messages, so that the consumer receiving the Interest-Return (NACK) can easily identify which path label is no longer valid.

A consumer may alternatively request that a forwarder detecting the inconsistency forward the Interest by means of normal LNPM FIB lookup rather than returning an error. The consumer endpoint, if it cares, can keep enough information about outstanding Interests to determine if the path label sent with the Interest fails to match the path label in the corresponding returned Content (Data), and use that information to replace stale path labels. It does so by setting the FALLBACK MODE flag of the path label TLV in its Interest message.

2.4. How to represent the Path Label

[Moiseenko2017] presents various options for how to represent a path label, with different tradeoffs in flexibility, performance and space efficiency. For this specification, we choose the `_Polynomial encoding_` which achieves reasonable space efficiency at the cost of establishing a hard limit on the length of paths that can be represented.

The polynomial encoding utilizes a fixed-size bit array. Each transit ICN forwarder is allocated a fixed sized portion of the bit array. This design allocates 12 bits (i.e. 4095 as a `_generator polynomial_`) to each intermediate ICN forwarder. This should match the scalability of today's commercial routers that support up to 4096 physical and logical interfaces and usually do not have more than a few hundred active ones.

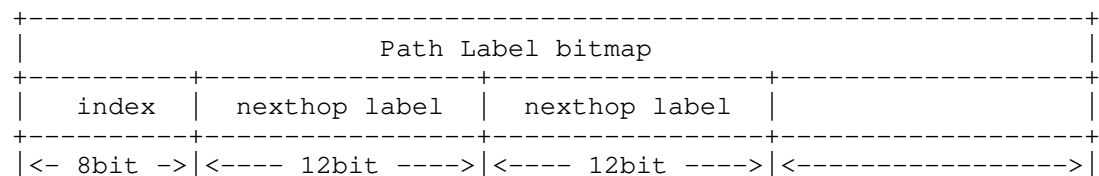


Figure 3: Fixed size path label

A forwarder that receives a Content (Data) message encodes the nexthop label in the next available slot and increments label index. Conversely, a forwarder that receives an Interest message reads the current nexthop label and decrements label index. Therefore, the

extra computation required at each hop to forward either an interest or Content Object message with a path label is minimized and constitutes a fairly trivial additional overhead compared to FIB lookup and other required operations.

This approach results in individual path label TLV instances being of fixed pre-computed size. While this places a hard upper bound on the maximum number of network hops that can be represented, this is not a significant a practical problem in NDN and CCNx, since the size can be pre-set during Content(Data) message encoding based on the exact number of network hops traversed by the Interest message. Even long paths of 24 hops will fit in a path label bitmap of 36 bytes if nexthop label is encoded in 12 bits.

3. Mapping to CCNx and NDN packet encodings

3.1. Path label TLV

A Path label TLV is the tuple: {[Flags], [Path Label Hop Count], [Nexthop Label], [Path label bitmap]}.

Flag	Value (hex)
DISCOVERY MODE	0x00
FALLBACK MODE	0x01
STRICT MODE	0x02

Table 1: Path label flags

The Path Label Hop Count (PLHC) MUST be incremented by NDN and CCNx forwarders if the Interest packet carries a path label and DISCOVERY mode flag is set. A producer node or a forwarder with cached data packet MUST use PLHC in calculation of a path label bitmap size suitable for encoding the entire path to the consumer. The Path Label Hop Count (PLHC) MUST be set to zero in newly created Data or Interest-Return (NACK) packets. A consumer node MUST reuse Path Label Hop Count (PLHC) together with the Path label bitmap (PLB) in order to correctly forward the Interest(s) along the corresponding network path.

If an NDN or CCNx forwarder supports path labeling, the Nexthop label MUST be used to determine the correct egress interface for an Interest packet carrying either the FALLBACK MODE or STRICT MODE flag. If any particular NDN or CCNx forwarder is configured to decrypt path labels of Interest packets (Section "Security considerations" (Section 5)), then the forwarder MUST

1. decrypt the path label with its own symmetric key,
2. update the nexthop label with outermost label in the path label,
3. decrement Path Label Hop Count (PLHC), and
4. remove the outermost label from the path label.

If any particular NDN or CCNx forwarder is NOT configured to decrypt path labels of Interest packets, then path label decryption SHOULD NOT be performed.

The Nexthop label MUST be ignored by NDN and CCNx forwarders if present in Data or Interest-Return (NACK) packets. If any particular NDN or CCNx forwarder is configured to encrypt path labels of Data and Interest-Return (NACK) packets (Section Security Considerations (Section 5)), then the forwarder MUST encrypt existing path label with its own symmetric key, append the nexthop label of the ingress interface to the path label, and increment Path Label Hop Count (PLHC). If any particular NDN or CCNx forwarder is NOT configured to encrypt path labels of Interest packets, then path label encryption SHOULD NOT be performed.

NDN and CCNx forwarders MUST fallback to longest name prefix match (LNPM) FIB lookup if an Interest packet carries an invalid nexthop label and the FALLBACK MODE flag is set.

CCNx forwarders MUST respond with an Interest Return packet specifying the T_RETURN_INVALID_PATH_LABEL code if Interest packet carries an invalid path label and the STRICT MODE flag is set.

CCNx forwarders MUST respond with an Interest Return packet specifying the T_RETURN_MALFORMED_INTEREST code if the Interest packet carries a path label TLV with both FALLBACK MODE and STRICT MODE flags set.

3.2. Path label encoding for CCNx

Path Label is an optional Hop-by-Hop header TLV that can be present in CCNx Interest, InterestReturn and Content Object packets.

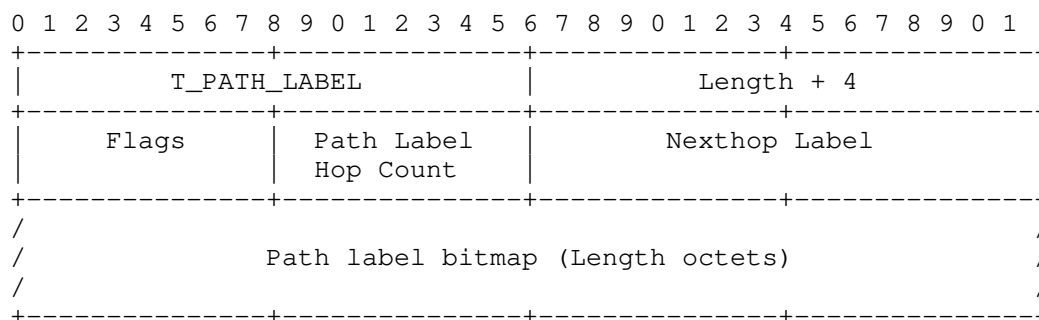


Figure 4: Path label Hop-by-Hop header TLV for CCNx

3.3. Path label encoding for NDN

Path Label is an optional TLV in NDN Interest, Data and NACK packets. The Path Label TLV SHOULD NOT take part in Interest, Data or NACK signature calculation as it is potentially modified at every network hop.

```

PathLabel = PATH-LABEL-TYPE TLV-LENGTH
           PathLabelFlags
           PathLabelBitmap

PathLabelFlags    = PATH-LABEL-FLAGS-TYPE
                   TLV-LENGTH ; == 1
                   OCTET

NexthopLabel      = PATH-LABEL-NEXTHOP-LABEL-TYPE
                   TLV-LENGTH ; == 2
                   2 OCTET

PathLabelHopCount = PATH-LABEL-HOP-COUNT-TYPE
                   TLV-LENGTH ; == 1
                   OCTET

PathLabelBitmap   = PATH-LABEL-BITMAP-TYPE
                   TLV-LENGTH ; == 64
                   64 OCTET

```

Figure 5: Path label TLV for NDN

Flag	Value (hex)
PATH-LABEL-TYPE	0x09
PATH-LABEL-FLAGS-TYPE	0x0B
PATH-LABEL-BITMAP-TYPE	0x0D
PATH-LABEL-NEXTHOP-LABEL-TYPE	0x0E
PATH-LABEL-HOP-COUNT-TYPE	0x0F

Table 2: TLV-TYPE number assignments

4. IANA Considerations

IANA is requested to make the following assignments:

1. Please assign the value 0x0004 for T_PATH_LABEL in the *CCNx Hop-by-Hop Types* registry.
2. Please assign the TLV types in Table 2 in the *CCNx Hop-by-Hop type registry*.
3. Please assign the value 0xA for the T_RETURN_INVALID_PATH_LABEL in the *CCNx Interest Return Code Types" registry*.
4. Please create the *CCNx Path Label Flags* registry and assign the values listed in Table 1.

5. Security Considerations

A path is invalidated by renumbering nexthop label(s). A malicious consumer can attempt to mount an attack by transmitting Interests with path labels which differ only in a single now-invalid nexthop label in order to _brute force_ a valid nexthop label. If such an attack succeeds, a malicious consumer would be capable of steering Interests over a network path that may not match the paths computed by the routing algorithm or learned adaptively by the forwarders.

When a label lookup fails, by default an `_Invalid path label_` Interest-Return (NACK) message is returned to the consumer. This contains a path label identical to the one included in the corresponding Interest message. A malicious consumer can therefore analyze the message's Hop Count field to infer which specific nexthop label had failed and direct an attack to influence path steering at that hop. This threat can be mitigated by the following countermeasures:

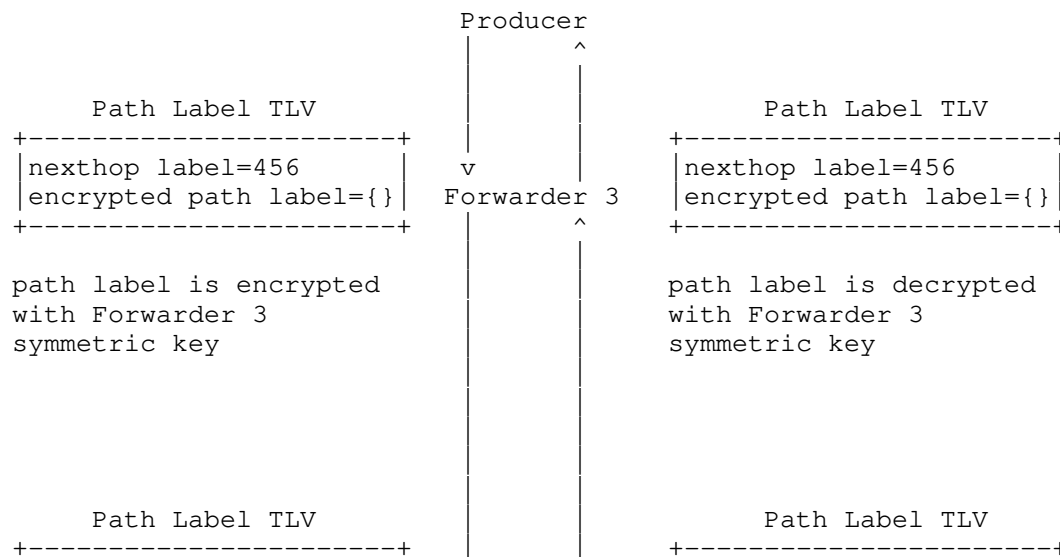
- * A nexthop label of larger size is harder to crack. If nexthop labels are not allocated in a predictable fashion by the routers, brute forcing a 32-bit nexthop label requires on average $O(2^{31})$ Interests. However, this specification uses nexthop labels with much less entropy (12 bits), so depending on computational hardness is not workable.
- * An ICN forwarder can periodically update nexthop labels to limit the maximum lifetime of paths. It is RECOMMENDED that forwarders update path labels at least every few minutes.
- * A void Hop Count field in an `_Invalid path label_` Interest-Return (NACK) message would not give out the information on which specific nexthop label had failed. An attacker might need to brute force all nexthop labels in all combinations. However, some useful diagnostic capability is lost by obscuring the hop count. For example the locus of routing churn is harder to pin down through analysis of path-steered pings or traceroutes. A forwarder MAY choose to invalidate the hop count in addition to changing nexthop labels periodically as above.

ICN protocols can be susceptible to a variety of cache poisoning attacks, where a colluding consumer and producer arrange for bogus content (with either invalid or inappropriate signatures) to populate forwarder caches. These are generally confined to on-path attacks. It is also theoretically possible to launch a similar attack without a cooperating producer such that the caches of on-path routers become poisoned with the content from off-path routers (i.e. physical connectivity, but no route in a FIB for a given prefix). We estimate that without any prior knowledge of the network topology, the complexity of this type of attack is in the ballpark of Breadth-First-Search and Depth-First-Search algorithms with the additional burden of transmitting 2^{31} Interests in order to crack a nexthop label on each hop. Relatively short periodic update of nexthop labels and anti-`_label scan_` heuristics implemented in the ICN forwarder may successfully mitigate this type of attack.

5.1. Cryptographic protection of a path label

If the countermeasures listed above do not provide sufficient protection against malicious mis-steering of Interests, the path label can be made opaque to the consumer endpoint via hop-by-hop symmetric cryptography applied to the path labels (Figure 6). This method is viable due to the symmetry of forward and reverse paths in CCNx and NDN architectures combined with ICN path steering requiring only reads/writes of the topmost nexthop label (i.e. active nexthop label) in the path label. This way a path steering capable ICN forwarder receiving a Data (Content) message encrypts the current path label with its own non-shared symmetric key prior to adding a new nexthop label to the path label. The Data (Content) message is forwarded downstream with unencrypted topmost (i.e active) nexthop label and encrypted remaining content of the path label. As a result, a consumer endpoint receives a Data (Content) message with a unique path label exposing only the topmost nexthop label as cleartext. A path steering forwarder receiving an Interest message performs label lookup using the topmost nexthop label, decrypts the path label with its own non-shared symmetric key, and forwards the message upstream.

Cryptographic protection of a path label does not require any key negotiation among ICN forwarders, and is no more expensive than MACsec or IPsec. It is also quite possible that strict hop-by-hop path label encryption is not necessary and path label encryption only on the border routers of the trusted administrative or routing domains may suffice.



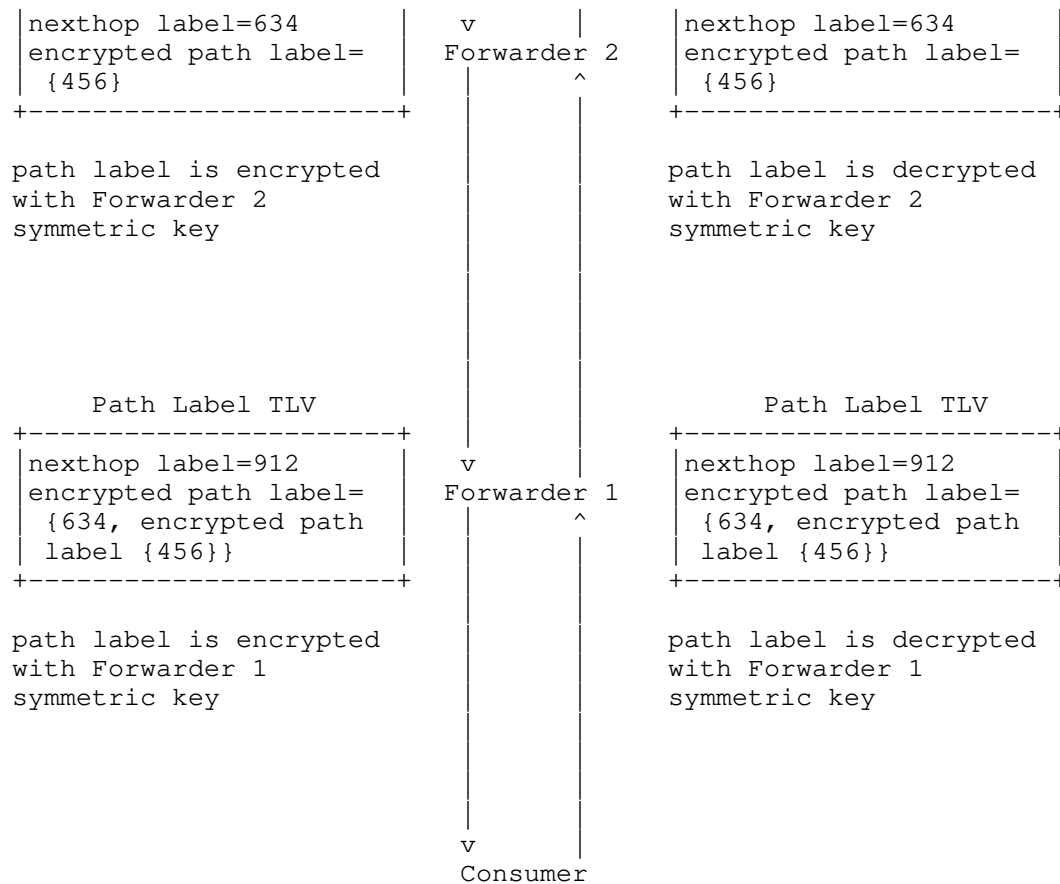


Figure 6: Path label protection with hop-by-hop symmetric cryptography

6. References

6.1. Normative References

[Moiseenko2017]

Moiseenko, I. and D. Oran, "Path Switching in Content Centric and Named Data Networks, in 4th ACM Conference on Information-Centric Networking (ICN 2017)", DOI 10.1145/3125719.3125721, September 2017, <<https://conferences.sigcomm.org/acm-icn/2017/proceedings/icn17-2.pdf>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <<https://www.rfc-editor.org/info/rfc8569>>.
- [RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <<https://www.rfc-editor.org/info/rfc8609>>.

6.2. Informative References

- [I-D.irtf-icnrg-flic]
Tschudin, C., Wood, C. A., Mosko, M., and D. Oran, "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-02, 4 November 2019, <<https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-flic-02>>.
- [I-D.irtf-icnrg-icnping]
Mastorakis, S., Gibson, J., Moiseenko, I., Droms, R., and D. Oran, "ICN Ping Protocol Specification", Work in Progress, Internet-Draft, draft-irtf-icnrg-icnping-02, 11 April 2021, <<https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-icnping-02>>.
- [I-D.irtf-icnrg-icntraceroute]
Mastorakis, S., Gibson, J., Moiseenko, I., Droms, R., and D. Oran, "ICN Traceroute Protocol Specification", Work in Progress, Internet-Draft, draft-irtf-icnrg-icntraceroute-02, 11 April 2021, <<https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-icntraceroute-02>>.
- [Mahdian2016]
Mahdian, M., Arianfar, S., Gibson, J., and D. Oran, "MIRCC: Multipath-aware ICN Rate-based Congestion Control, in Proceedings of the 3rd ACM Conference on Information-Centric Networking", DOI 10.1145/2984356.2984365, 2022, <<http://conferences2.sigcomm.org/acm-icn/2016/proceedings/p1-mahdian.pdf>>.
- [NDN] "Named Data Networking", various, <<https://named-data.net/project/execsummary/>>.

- [NDNTLV] "NDN Packet Format Specification.", 2016,
<<http://named-data.net/doc/ndn-tlv/>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8793] Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): Content-Centric Networking (CCNx) and Named Data Networking (NDN) Terminology", RFC 8793, DOI 10.17487/RFC8793, June 2020, <<https://www.rfc-editor.org/info/rfc8793>>.
- [Song2018] Song, J., Lee, M., and T. Kwon, "SMIC: Subflow-level Multi-path Interest Control for Information Centric Networking, in 5th ACM Conference on Information-Centric Networking", DOI 10.1145/3267955.3267971, 2018, <<https://conferences.sigcomm.org/acm-icn/2018/proceedings/icn18-final62.pdf>>.

Authors' Addresses

Ilya Moiseenko
UCLA
Email: iliamo@ucla.edu

Dave Oran
Network Systems Research and Design
4 Shady Hill Square
Cambridge, MA 02138
United States of America
Email: daveoran@orandom.net

ICNRG
Internet-Draft
Intended status: Informational
Expires: May 2, 2020

D. Trossen
C. Wang
S. Robitzsch
InterDigital Inc.
M. Reed
M. Al-Naday
Essex University
J. Riihijarvi
RWTH Aachen
October 30, 2019

Internet Services over ICN in 5G LAN Environments
draft-trossen-icnrg-internet-icn-5glan-00

Abstract

In this draft, we provide architecture and operations for enabling Internet services over ICN over (5G-enabled) LAN environments. Operations include ICN API to upper layers, HTTP over ICN, Service Proxy Operations, ICN Flow Management, Name Resolution, Mobility Handling, and Dual Stack Device Support.

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 <https://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 May 2, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Use Cases	5
3.1. 5G Control Plane Services	5
3.2. HTTP-based Streaming	6
4. 5GLAN in 5G Next Generation Core Network Architecture	7
4.1. Realization in SDN Transport Networks	10
4.2. Realization in Other Transport Networks	10
5. Internet Services over ICN over 5GLAN	11
5.1. General Operations	13
5.2. ICN API to Upper Layers	14
5.3. HTTP over ICN	15
5.3.1. General Mapping Procedures	15
5.3.2. Realizing Ad-Hoc Multicast Responses for HTTP	15
5.4. Service Proxy Operations	16
5.5. ICN Flow Management	17
5.6. NR Operations	17
5.7. Mobility Handling	17
5.8. Dual Stack Device Support	17
6. Deployment Considerations	18
7. Conclusion	19
8. IANA Considerations	19
9. Security Considerations	19
10. Acknowledgments	19
11. Informative References	19
Authors' Addresses	22

1. Introduction

As discussed in [I-D.ravi-icnrg-5gc-icn], Information-Centric Networks (ICN) could be more easily implemented in a Local Area Network (LAN) environment. In relation to 5G, this specifically would realize an ICN deployment without requiring integration of ICN capabilities into the 5G core network itself.

In the currently defined 5G core network, 5GLAN capabilities are being introduced that provide a LAN abstraction to 5G endpoints, allowing for Ethernet packets to be sent across a 5G network,

therefore extending the provisioning of LAN capabilities from fixed and Wifi-based networks to cellular ones.

Utilizing such ICN realization over 5GLAN, the objective of this draft is to propose an architecture to enable Internet services over such ICN-over-LAN environment with the reference architectural discussions in the 5G core network 3GPP specifications [TS23.501] [TS23.502] forming the basis of our discussions. This draft also complements work related to various ICN deployment opportunities explored in [I-D.irtf-icnrg-deployment-guidelines], where 5G technology is considered as one of the promising alternatives. In that, ICN is used as an underlay technology to provide routing capabilities to Internet services.

Through such replacement of IP routing with ICN routing, we capitalize on several ICN capabilities:

- o Edge Computing: Multi-access Edge Computing (MEC) is located at the edge of the network and aids several latency sensitive applications such as augmented and virtual reality (AR/VR), as well as the ultra reliable and low latency class (URLLC) of applications such as autonomous vehicles. Enabling edge computing over an IP converged 5GC comes with the challenge of application level reconfiguration required to re-initialize a session whenever it is being served by a non-optimal service instance topologically. In contrast, named-based networking, as considered by ICN, naturally supports service-centric networking, which minimizes network related configuration for applications and allows fast resolution for named service instances. This opportunity is realized by interpreting Internet services as transactions over an ICN routed network with flexible routing to the nearest execution point for said transaction.
- o Edge Storage and Caching: A principal design feature of ICN is the secured content (or named data) object, which allows location independent data replication at strategic storage points in the network, or data dissemination through ICN routers by means of opportunistic caching. These features benefit both real-time and non-real-time applications whenever there is spatial and temporal correlation among content accessed by these users, thereby advantageous to both high-bandwidth and low-latency applications such as conferencing, AR/VR, and non-real time applications such as Video-on-Demand (VOD) and IoT transactions. This opportunity is realized by the transaction-based model of realizing Internet service on top of an ICN routed network, where transaction results can be retrieved from a number of network locations.

- o **Opportunistic Multicast:** The vast majority of current Internet traffic is due to unicast delivery of relatively immutable content such as video or software to very large client groups. This has resulted in large amount of redundancy in network traffic, as well as creating capacity bottlenecks both in the core network as well as the server infrastructure serving the content. Technologies such as content delivery networks (CDNs) help to spread out the network load, but are complex to manage, have inherent limits in terms of how rapidly they can react to changing network and server conditions, and cannot fundamentally reduce the network overhead arising from redundant unicast streams. Furthermore, CDNs traditionally only reach into Points-of-Presence (POP) within customer networks, therefore not reducing the load of transfer from said POP to the end customers in that edge network. In contrast, ICN enables opportunistic multicast delivery of content. We realize this opportunity by automatically delivering responses to quasi-concurrent requests in a single lightweight multicast transmission over the L2 customer network, extending the reach of CDNs down to the end user. Unlike traditional IP multicast, no setup time overhead is added and no per-flow state is required in the network.

In this document, we first outline possible use cases, capitalizing on the aforementioned ICN capabilities before discussing the proposed extensions to 5G to support a cellular-based LAN connectivity before outlining our proposal to support Internet services over an ICN-routed LAN connectivity in such 5G environments.

2. Terminology

Following are terminologies relevant to this draft:

5G-NextGen Core (5GC): Refers to the new 5G core network architecture being developed by 3GPP, we specifically refer to the architectural discussions in [TS23.501] [TS23.502].

5GLAN: Refers to the extensions to the new 5G core network architecture that provide LAN connectivity to 5G devices connected via, e.g., new 5G air interfaces.

User Plane Function (UPF): UPF is the generalized logical data plane function with context of the UE PDU session. UPFs can play many roles, such as, being a flow classifier, a PDU session anchoring point, or a branching point.

Packet Data Network (PDN or DN): This refers to service networks that belong to the operator or third party offered as a service to the UE.

Unified Data Management (UDM): Realizes unified data management for wireless, wireline and any other types of subscribers for M2M, IOT applications, etc. UDM reports subscriber related vital information e.g. virtual edge region, list of location visits, sessions active etc. UDM works as a subscriber anchor point so that means OSS/BSS systems will have centralized monitoring-of/access-to of the system to get/set subscriber information.

Authentication Server Function (AUSF): Provides mechanism for unified authentication for subscribers related to wireless, wireline and any other types of subscribers such as M2M and IOT applications. The functions performed by AUSF are similar to HSS with additional functionalities to related to 5G.

Session Management Function (SMF): Performs session management functions for attached users equipment (UE) in the 5G Core. SMF can thus be formed by leveraging the Control and User Plane Separation (CUPS) feature with control plane session management.

Access Mobility Function (AMF): Perform access mobility management for attached user equipment (UE) to the 5G core network. The function includes, network access stratus (NAS) mobility functions such as authentication and authorization.

Application Function (AF): Helps with influencing the user plane routing state in 5GC considering service requirements.

Network Slicing: This conceptualizes the grouping for a set of logical or physical network functions with its own or shared control, data and service plane to meet specific service requirements.

3. Use Cases

3.1. 5G Control Plane Services

We exemplify the need for chaining service functions at the level of a service name through a use case stemming from the current 3GPP Rel-16 work on Service Based Architecture (SBA) [TS29.500] [SBA-ENHANCEMENT]. In this work, mobile network control planes are proposed to be realized by replacing the traditional network function interfaces with a fully service-based one. HTTP was chosen as the application layer protocol for exchanging suitable service requests [TS29.500]. With this in mind, the exchange between, say the 3GPP (Rel-15) defined Session Management Function (SMF) and the Access and Mobility management Function (AMF) in a 5G control plane is being described as a set of web service like requests which are in turn embedded into HTTP requests. Hence, interactions in a 5G control

plane can be modelled based on service function chains where the relationship is between the specific service function endpoints that implement the necessary service endpoints in the SMF and AMF. The service functions are exposed through URIs with work ongoing to define the used naming conventions for such URIs.

This move from a network function model (in pre-Rel 15 systems of 3GPP) to a service-based model is motivated through the proliferation of data center operations for mobile network control plane services. In other words, typical IT-based methods to service provisioning, in particular that of virtualization of entire compute resources, are envisioned to be used in future operations of mobile networks. Hence, operators of such future mobile networks desire to virtualize service function endpoints and direct (control plane) traffic to the most appropriate current service instance in the most appropriate (local) data centre, such data centre envisioned as being interconnected through a software-defined wide area network (SD-WAN). 'Appropriate' here can be defined by topological or geographical proximity of the service initiator to the service function endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data center centric operation is extended with the trend towards regionalization of load through a 'regional office' approach, where micro data centers provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the 'most appropriate' service endpoint for a particular incoming service request. This 5G control plane scenario capitalizes on the edge computing capabilities of ICN by allowing for fast redirections of HTTP-based transactions to the nearest control plane service realization within the distributed data centre of the 5G operator infrastructure.

3.2. HTTP-based Streaming

With the extensive use of "web technology", "distributed services" and availability of heterogeneous network, HTTP has effectively transitioned into the common transport or session layer for E2E and multi-hop communication across the web. Assume clients that are consuming the same content (such as a TV program) and that this content has for each block (typically segments worth 2 seconds of content) a set of outstanding requests from its clients. HTTP request and response used in media streaming services like HLS, use HTTP response for delivery of content. In such scenarios, where semi-synchronous access to the same resource occurs (such as watching prominent videos over Netflix or similar platforms or live TV over HTTP), traffic grows linearly with the number of viewers since the HTTP-based server will provide an HTTP response to each individual

viewer. To mitigate the load impact, operators often utilize IP multicast underneath HTTP (for live TV) to create fewer, multicast, streams; though this comes with the high flow setup and management cost. This poses a significant burden on operators in terms of costs and on users in terms of likely degradation of quality.

This problem is not limited to traditional TV broadcasting. Consider a virtual reality use case where several users are joining a VR session at the same time, e.g., centered around a joint event. Hence, due to the temporal correlation of the VR sessions, we can assume that multiple requests are sent for the same content at any point, particularly when viewing angles of VR clients are similar or the same. Due to availability of virtual functions and cloud technology, the actual end point from where content is delivered may change. For this type of scenarios, the opportunistic multicast capability of ICN may be utilized to reduce overall load in the network, as well as on the server providing the HTTP responses. The latter also allows constrained resources to serve a higher volume of demands and therefore incur a higher impact on traffic distribution in the network.

4. 5GLAN in 5G Next Generation Core Network Architecture

In this section, for brevity purposes, we restrict the discussions to the 5G extensions currently studied in 3GPP to facilitate a distributed, cellular-based LAN connectivity to end users, based on the 5G next generation core network architecture. For more information on the latter, we refer to [TS23.501] [TS23.502] as well as [I-D.ravi-icnrg-5gc-icn].

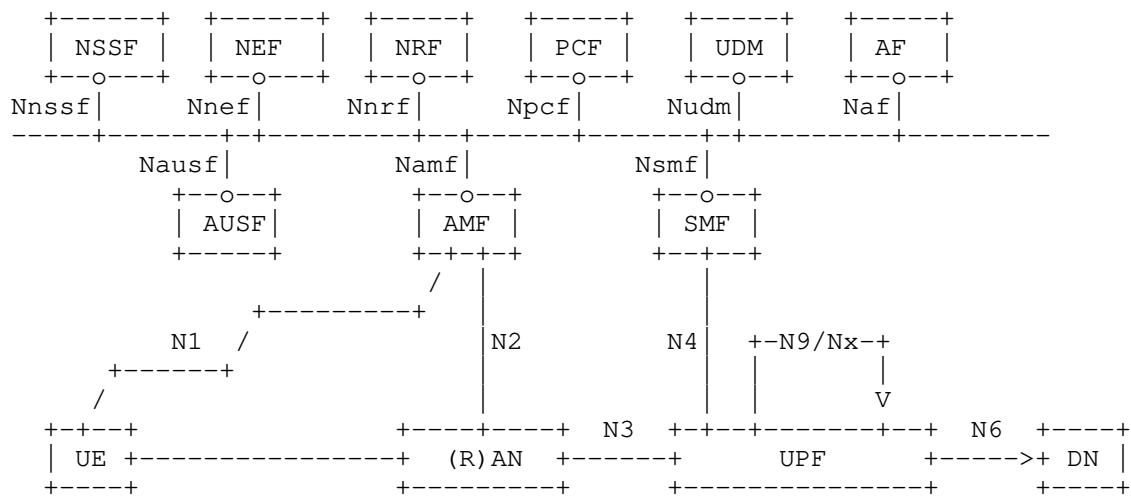


Figure 1: 5G Core Network with Vertical_LAN (5GLAN) Extensions

Figure 1 shows the current 5G Core Network Architecture being discussed within the scope of the normative work addressing 5GLAN Type services in the 3GPP System Architecture Working Group 2 (3GPP SA2), referred formally as "5GS Enhanced support of Vertical and LAN Services" [SA2-5GLAN]. The goal of this work item is to provide distributed LAN-based connectivity between two or more terminals or User Equipment entities (UEs) connected to the 5G network. The SMF (session management function) provides a registration and discovery protocol that allows UEs wanting to communicate via a relevant 5GLAN group towards one or more UEs also members of this 5GLAN group, to determine the suitable forwarding information after each UE previously registered suitable identifier information with the SMF responsible to manage the paths across UEs in a 5GLAN group. UEs register and discover (obtain) suitable identifiers during the establishment of a Protocol Data Unit (PDU) Session or PDU Session Modification procedure. Suitable identifier information, according to [SA2-5GLAN], are Ethernet MAC addresses as well as IP addresses (the latter is usually assigned during the session setup through the SMF, i.e., the session management function).

The SMF that manages the path across UEs in a 5GLAN group, then establishes the suitable procedures to ensure the forwarding between the required UPFs (user plane functions) to ensure the LAN connectivity between the UEs (user equipments) provided in the original request to the SMF. When using the N9 interface to the UPF, this forwarding will rely on a tunnel-based approach between the UPFs along the path, while the Nx interface uses path-based forwarding

between UPFs, while LAN-based forwarding is utilized between the final UPF and the UE (utilizing the N3 interface towards the destination UE).

In the following, path-based forwarding is assumed, i.e., the usage of the Nx interface and the utilization of a path identifier for the end-to-end LAN communication. Here, the path between the source and destination UPFs is encoded through a bitfield, provided in the packet header. Each bit position in said bitfield represents a unique link in the network. Upon receiving an incoming packet, each UPF inspects said bitfield for the presence of any local link that is being served by one of its output ports. Such presence check is implemented via a simple binary AND and CMP operation. If no link is being found, the packet is dropped. Such bitfield-based path representation also allows for creating multicast relations in an ad hoc manner by combining two or more path identifiers through a binary OR operation. Note that due to the assignment of a bit position to a link, path identifiers are bidirectional and can therefore be used for request/response communication without incurring any need for path computation on the return path.

For sending a packet from one Layer 2 device (UE) connected to one UPF (via a RAN) to a device connected to another UPF, we provide the MAC address of the destination and perform a header re-write by providing the destination MAC address of the ingress UPF when sending from source device to ingress and placing the end destination MAC address in the payload. Upon arrival at the egress UPF, after having applied the path-based forwarding between ingress and egress UPF, the end destination address is restored while the end source MAC is placed in the payload with the egress L2 forwarder one being used as the L2 source MAC for the link-local transfer. At the end device (or proxy device), the end source MAC address is restored as the source MAC, providing an abstraction of a link-local L2 communication between the end source and destination devices.

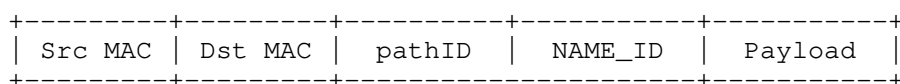


Figure 2: General Packet Structure

For this end-to-end transfer, the general packet structure of Figure 2 is used. The Name_ID field is being used for the ICN operations, while the payload contains the information related to the transaction-based flow management described in Section 5.6 and the

PATH_ID is the bitfield-based path identifier for the path-based forwarding.

4.1. Realization in SDN Transport Networks

An emerging technology for Layer 2 forwarding that suits the 5GLAN architecture in Figure 1 is that of Software-Defined networking (SDN) [SDN-DEFINITION], which allows for programmatically forwarding packets at Layer 2. Switch-based rules are being executed with such rules being populated by the SDN controller. Rules can act upon so-called matching fields, as defined by the OpenFlow protocol specification [OpenFlowSwitch]. Those fields include Ethernet MAC addresses, IPv4/6 source and destination addresses and other well-known Layer 3 and even 4 transport fields.

As shown in [Reed], efficient path-based forwarding can be realized in SDN networks by placing the aforementioned path identifiers into the IPv6 source/destination fields of a forwarded packet. Utilizing the IPv6 source/destination fields allows for natively supporting 256 links in a transport network. Larger topologies can be supported by extension schemes but are left out of this paper for brevity of the presentation. During network bootstrapping, each link at each switch is assigned a unique bitnumber in the bitfield. In order to forward based on such bitfield path information, the SDN controller is instructed to insert a suitable wildcard matching rule into the SDN switch. This wildcard at a given switch is defined by the bitnumber that has been assigned to a particular link at that switch during bootstrapping. Wildcard matching as a generalization of longest prefix matching is natively supported by SDN-based switches since the OpenFlow v1.3 specification, efficiently implemented through TCAM based operations. With that, SDN forwarding actions only depend on the switch-local number of output ports, while being able to transport any number of higher-layer flows over the same transport network without specific flow rules being necessary. This results in a constant forwarding table size while no controller-switch interaction is necessary for any flow setup; only changes in forwarding topology (resulting in a change of port to bit number assignment) will require suitable changes of forwarding rules in switches.

4.2. Realization in Other Transport Networks

Although we focus the methods in this draft on Layer 2 forwarding approaches and realization of Internet-over-ICN over a 5G LAN enabled network, path-based transport networks can also be established as an overlay over otherwise Layer 2 networks. For instance, the BIER (Bit Indexed Explicit Replication) [RFC8279] efforts within the Internet Engineer Task Force (IETF) establish such path-based forwarding

transport as an overlay over existing, e.g., MPLS networks. The path-based forwarding identification is similar to the aforementioned SDN realization although the bitfield represents ingress/egress information rather than links along the path.

Yet another transport network example is presented in [Khalili], utilizing flow aggregation over SDN networks. The flow aggregation again results in a path representation that is independent from the specific flows traversing the network.

The proposed traffic engineering extensions to BIER, presented in [I-D.ietf-bier-te-arch], directly align with the SDN-based realization presented in Section 4.1, by proposing the same bitposition per transport link assignment being used, resulting in BIER bitstrings in which a dedicated forwarding path is encoded as a unique bitpattern containing said bitpositions of the chosen forwarding links. The BIER-TE controller plays a similar role as the northbound SDN controller application utilized for the solution in Section 4.1.

5. Internet Services over ICN over 5GLAN

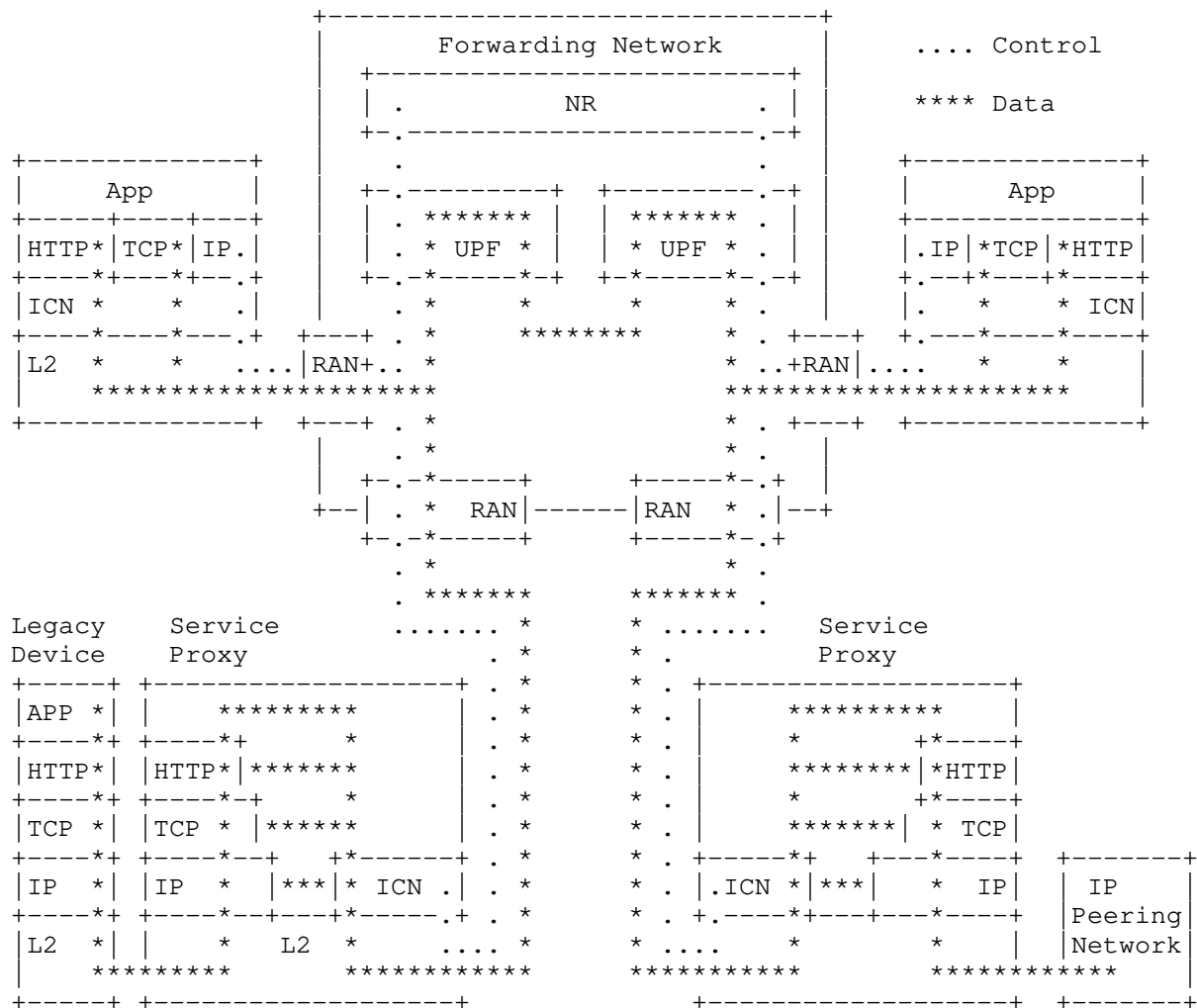


Figure 3: Internet Services over ICN over 5GLAN

Figure 3 shows the protocol layering for realizing Internet protocols over an ICN over 5GLAN transport, assuming an end-to-end LAN connectivity provided by solutions such as 5GLAN.

Note that such LAN connectivity can also be found in environments such as localized LAN-based deployments in smart cities, enterprises and others, with the UPF representing, e.g., an SDN switch (utilizing the methods outlined in Section 4.1). Hence, the solutions described in this section also applies to those deployments.

Key to the approach is that Internet services are being interpreted as the main unit of transfer in the architecture shown in Figure 3. For this, any Internet service is treated as a Named Service Transaction (NST) which is in turn suitably routed over an ICN layer in one or more other devices. As a result of this name-based interpretation of any Internet service, the protocol stack in end devices flattens to four layers with Internet services and ICN, with ICN acting as a name-based routing layer for all IP protocols implemented atop, with Layer 1 and 2 realizing the end-to-end packet forwarding outlined in Section 4 (over a 5G environment) or a general LAN environment provided through WiFi or fixed Ethernet technologies.

The general ICN operations are presented in Section 5.1 before discussing the assumed (strawman) API to the ICN layer in Section 5.2, which is used in turn to define the mapping of HTTP transactions to operations at the ICN layer in Section 5.3 for the example of HTTP. As explained in that section, the ICN layer uses an interaction with the NR to register and discover HTTP-based services for determining the suitable end-to-end packet forwarding information.

Interfaces to legacy devices and peering networks are preserved through service proxy devices, which terminate a traditional Internet protocol stack communication and translate it into a resulting flat protocol transaction. Termination here can be based on well-known port numbers for specific Internet protocols, ultimately falling back to the IP datagram service being the minimal service being mapped. The operations of said service proxy devices is described in Section 5.4.

An important aspect of the architecture is the mapping of the end-to-end flow semantic established in many Internet services onto the flat protocol stack. Section 5.5 outlines the flow management that exists between the end devices.

The mapping of protocol identifiers onto ICN forwarding relations, i.e., the operations of the name resolver (NR), shown in Figure 3, is described in Section 5.6, followed by the procedures for handling mobility of service providers and consumers in Section 5.7. Finally, the support for dual-stack devices, not requiring a service proxy device yet being able to also connect to existing IP routed networks, is described in Section 5.8.

5.1. General Operations

The semantics of our name-based routing is that of a publish-subscribe system over a name. The intention to receive packets with a certain name is expressed through a subscription while sending

packets to a name is expressed through a publication. The matching of a sender to a receiver is realized through NR in Figure 3. The exact nature of the matching is defined through the semantics of the service and, therefore, through the nature of the name provided. For instance, HTTP and raw Internet services are matched to exactly one subscriber only, providing an anycast capability, while IP multicast services are matched against any subscriber (with the IP multicast address being the name).

Structured names are used with the root specific to the (Internet) service name, such as URL, and therefore deriving the matching semantics directly from the name.

The subscription to a name is realized through a registration protocol between end device and NR. Hence, any end device exposing a certain Internet service registers the suitable name with the NR, which in turn stores reachability information that is suitable for path calculation between the ingress and egress L2 forwarders between which the communication eventually will take place. In our current realization, we utilize shortest paths only although other link weights can be utilized for, e.g., delay-constrained and other policies.

In our realization, we use network domain unique host identifiers that are being assigned to end devices during the connectivity setup. Sending a packet of a given Internet service is realized through a discovery protocol, which returns a suitable pathID, i.e., the forwarding information between ingress and egress L2 forwarder, and the destination MAC address of the hosting end device. It is this pathID and MAC address that is being used in the general packet structure of Figure 2 to forward the packet to the destination.

To reduce latency in further communication, the forwarding information is locally cached at the end device, while the cached information is being maintained through path updates sent by the NR in case of hosting end devices having moved or de-registered, therefore avoiding stale forwarding information.

5.2. ICN API to Upper Layers

The operations of the ICN layer are exposed to upper layers in Figure 1 through the following API calls, being exemplary here for the further explanation of operations in the next sub-sections:

- o conn = send(name, payload)
- o send(conn, payload)

- o conn = receive(name, &payload)
- o receive(conn, &payload)

The first send() call is used for initiating a send operation to a name with a connection handle returned, while the second send() is used for return calls, using a connection parameters that is being received with the receive() call to an incoming connection or for subsequent outgoing calls after an initial request to a name has been made. A return send() is being received at the other (client) side through the second receive() call where the conn parameter is obtained by the corresponding send() call for the outgoing call. With these API functions, we provide means for providing name-based transactions with return responses association provided natively.

The conn parameter represents the bitfield used for path-based forwarding in the remote host case or the hash of the local MAC address in case of link-local connections.

5.3. HTTP over ICN

5.3.1. General Mapping Procedures

To realize the flat device nature, Internet service layers, such as the HTTP protocol stack or the TCP protocol stack, will need to be adapted to run atop this new API, implementing the semantics of the respective Internet protocol through suitable transactions at the name level. In the example of HTTP, the standard operations of DNS resolution for the server to be contacted and opening of a TCP socket are altogether replaced by a single send(FQDN, HTTP request) call, while the response will be sent by the server, which received the request through a receive(FQDN, &payload) call, using the returned conn parameter to send the response with the second send() API call. Note that the use of bidirectional pathIDs, no NR lookup is performed at the HTTP serving endpoint.

5.3.2. Realizing Ad-Hoc Multicast Responses for HTTP

The basis of a named service transaction allows to deliver the same HTTP responses to several requestees in efficient multicast (see [I-D.ietf-bier-multicast-http-response] for use cases in a BIER-based transport network environment).

This opportunity is realized by sending the same payload (i.e., an HTTP response to the same resource across a number of pending requests) through a combination of several conn parameters received in the incoming requests via the receive() function.

What is required in the HTTP stack implementation is a logic to decide that two or more outstanding requests are possible to be served by one response. For this, upon receiving an incoming request, the HTTP stack determines any outstanding request to the same resource. 'Same' here is defined as URI-specific combination of the request URI and URI-specific header fields, such as browsing agent or similar, called requestID in the following.

Once such determination is made that two requests are relating to the same resource, i.e., are having the same request ID, the HTTP stack maintains a temporary mapping of the request ID to the respective conn parameters delivered by the receive() call. Upon receiving the HTTP response from its application-level logic, the HTTP stack will generate the suitable send(conn, payload) call where the provided conn parameter is bitwise OR of all previously stored conn parameters received in the receive() call. The ICN layer will recognize the use of those ad-hoc created conn parameters and set the destination MAC address in the general packet structure of Figure 2 to the Ethernet broadcast MAC address as the destination address, leading to sending the response to all end devices at the egress L2 forwarders to which the response will be forwarded based on the combined conn parameter. Alternatively, one could request IEEE assignment for a specific Ethernet multicast address for this scheme instead of using the broadcast address. For the local end device to determine the relevance of the response received at the broadcast channel, the HTTP stack of the serving endpoint includes the aforementioned requestID into the payload of the packet (see Figure 2), while the originating endpoint maintains an internal table with the requestID of pending requests and its associated conn handle. If no matching requestID is found, the packet is not being delivered to the ICN layer of the incoming device. If a request is found, the ICN layer delivers the response via the receive() call, using the conn handle stored in the pending request table. Note that this filtering mechanism can easily be implemented in hardware upon standardizing the appropriate payload and header fields.

5.4. Service Proxy Operations

The service proxy in Figure 3 serves the integration of legacy devices, i.e., with regular IP protocol stack, and the interconnection to IP-based peering networks. It registers suitable identifiers with the NR to ensure the reception of (ICN) packets, while providing suitable protocol termination for the various supported protocols. For instance, for HTTP, the service proxy towards the peering network will register a wildcard name to the NR to receive any HTTP request not destined to a network-locally registered FQDN, operating as an HTTP proxy towards the peering network.

5.5. ICN Flow Management

EDITOR NOTE: left for future draft updates.

5.6. NR Operations

The NR in Figure 3 combines the operations of the SMF and the PMF in 5GLAN (see Figure 1), by allowing for registering IP protocol identifiers for discovery and subsequent path computation by resolving the destination(s) to a suitable pathID and destination MAC address for forwarding. This will require extensions to the operations of the SMF to allow for such higher layer identifiers to be registered (and discovered), in addition to the already supported Ethernet and IP addresses.

5.7. Mobility Handling

EDITOR NOTE: left for future draft updates.

5.8. Dual Stack Device Support

Figure 3 outlines a protocol stack for the user equipment that realizes Internet services on top of the proposed name-based routing layer as a single stack device. However, [I-D.irtf-icnrg-icn-lte-4g] outlines the possibility of supporting dual-stack devices for 4G LTE networks by allowing IP as well as ICN protocol stacks to be deployed with the operation of IP and ICN based applications. [I-D.ravi-icnrg-5gc-icn] outlines the same dual-stack device realization for a 5G ICN realization. For both environments, a convergence layer is described that selects the appropriate data path for each ICN or IP application, e.g., based on configuration per application (similar to selecting network interfaces such as WiFi over cellular).

As a possible data path selection, [I-D.irtf-icnrg-icn-lte-4g] and [I-D.ravi-icnrg-5gc-icn] envision the realization of Internet-over-ICN (Section 4.2 in [I-D.irtf-icnrg-icn-lte-4g]) in which the convergence layer would realize similar mapping functions as described in this draft. Hence, we foresee the utilization of such dual-stack devices connected to an Internet services over ICN over 5GLAN environment. When utilizing the service proxy, IP applications that are configured to use the IP data path only could still utilize the ICN-based forwarding in the network. In that case, functionality such as the opportunistic multicast in Section 5.3.2 would only reach up to the service proxy with unicast traffic continuing along the data path towards the user equipment.

6. Deployment Considerations

The work in [I-D.irtf-icnrg-deployment-guidelines] outlines a comprehensive set of considerations related to the deployment of ICN. We now relate the solutions proposed in this draft to the two main aspects covered in the deployment considerations draft, namely the 'deployment configuration' (covered in Section 3 of [I-D.irtf-icnrg-deployment-guidelines]) that is being realized and the 'deployment migration paths' (covered in Section 4 of [I-D.irtf-icnrg-deployment-guidelines]) that are being provided.

The solutions proposed in this draft relate to those "deployment configuration" as follows:

- o The realization of Internet service on top of an ICN routing capabilities, as proposed in Section 5, follows the "ICN-as-an-Underlay" categorization, interpreting the ICN routing as an underlay to the Internet services with the path-based forwarding being compatible with the 5GLAN forwarding capabilities currently discussed in 3GPP and therefore providing an underlay integration capability for the ICN forwarding used in the proposed solution.
- o The deployment of 5GLAN based ICN capabilities can be realized following the "ICN-as-a-Slice" deployment configuration, i.e., the 5GLAN connectivity is provided to a "vertical 5G customer" which in turn provides the ICN capability over 5GLAN within said network (and compute) slice at the endpoints of the 5GLAN connectivity, as proposed in Section 3.

In relation of the 'deployment migration paths', the solutions in this draft relate as follows:

- o The integration with the 5GLAN capability, as proposed in Section 5, facilitates "edge network migration" (interpreting the cellular sub-system here as an edge network albeit a possibly geographically large one).
- o The single stack realization, as proposed in Figure 3, as well as the dual-stack deployment, as proposed in Section 5.8, facilitate "application and services migration" through not only supporting ICN applications but also Internet applications through the proposed Internet-over-ICN mapping in the terminal.
- o The Internet over ICN over 5GLAN deployment, possibly combined with an ICN-as-a-Slice deployment, facilitates the "content delivery networks migration" through a deployment of Internet-over-ICN-based 5GLAN connected CDN elements in (virtualized) edge network nodes or POP locations in the customer (5G) network.

7. Conclusion

In this draft, we explored the feasibility of enabling Internet services directly over ICN network over (5G)LAN environments. We proposed the architecture and discussed corresponding operations of mapping Internet services onto name-based transactions, with the specific example of HTTP-based transactions. We described the flow management, the realization of opportunistic multicast responses for HTTP as well as the realization of dual-stack user equipment. Future updates to the draft will provide more details to the flow management in terms of reliable transport between Internet-overIP-ICN enabled end-systems as well as mobility handling. We also described the deployment scenario for supporting Internet services over ICN over 5GLAN.

8. IANA Considerations

This document requests no IANA actions.

9. Security Considerations

Editor Note: to be added in future drafts.

10. Acknowledgments

Work towards developing the solutions outlined in this draft have been funded under grants of the [H2020POINT] and [H2020FLAME] projects.

11. Informative References

[H2020FLAME]

H2020, "The FLAME Project", <https://www.ict-flame.eu/> .

[H2020POINT]

H2020, "The POINT Project", <https://www.point-h2020.eu/> .

[I-D.galis-anima-autonomic-slice-networking]

Galis, A., Makhijani, K., Yu, D., and B. Liu, "Autonomic Slice Networking", draft-galis-anima-autonomic-slice-networking-05 (work in progress), September 2018.

[I-D.ietf-bier-multicast-http-response]

Trossen, D., Rahman, A., Wang, C., and T. Eckert, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", draft-ietf-bier-multicast-http-response-01 (work in progress), June 2019.

- [I-D.ietf-bier-te-arch]
Eckert, T., Cauchie, G., and M. Menth, "Traffic Engineering for Bit Index Explicit Replication (BIER-TE)", draft-ietf-bier-te-arch-04 (work in progress), October 2019.
- [I-D.irtf-icnrg-deployment-guidelines]
Rahman, A., Trossen, D., Kutscher, D., and R. Ravindran, "Deployment Considerations for Information-Centric Networking (ICN)", draft-irtf-icnrg-deployment-guidelines-07 (work in progress), September 2019.
- [I-D.irtf-icnrg-icn-lte-4g]
suthar, P., Stolic, M., Jangam, A., Trossen, D., and R. Ravindran, "Native Deployment of ICN in LTE, 4G Mobile Networks", draft-irtf-icnrg-icn-lte-4g-04 (work in progress), September 2019.
- [I-D.muscariello-intarea-hicn]
Muscariello, L., Carofiglio, G., Auge, J., and M. Papalini, "Hybrid Information-Centric Networking", draft-muscariello-intarea-hicn-03 (work in progress), October 2019.
- [I-D.ravi-icnrg-5gc-icn]
Ravindran, R., suthar, P., Trossen, D., Wang, C., and G. White, "Enabling ICN in 3GPP's 5G NextGen Core Architecture", draft-ravi-icnrg-5gc-icn-04 (work in progress), May 2019.
- [I-D.white-icnrg-ipoc]
White, G., Shannigrahi, S., and C. Fan, "Internet Protocol Tunneling over Content Centric Mobile Networks", draft-white-icnrg-ipoc-02 (work in progress), June 2019.
- [Khalili] Khalili, R., Poe, W., Despotovic, Z., and A. Hecker, "Reducing State of SDN Switches in Mobile Core Networks by Flow Rule Aggregation", IEEE ICCCN 2016, Hawaii, USA, August 2016.
- [OpenFlowSwitch]
Open Networking Foundation, available at <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>, "OpenFlow Switch Specification V1.5.1", 2018.

- [Reed] Reed, M., AI-Naday, M., Thomos, N., Trossen, D., Petropoulos, G., and S. Spirou, "Stateless Multicast Switching in Software Defined Networks", IEEE ICC 2016, Kuala Lumpur, Malaysia, 2016.
- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [SA2-5GLAN] 3gpp-5glan, "SP-181129, Work Item Description, Vertical_LAN(SA2), 5GS Enhanced Support of Vertical and LAN Services", 3GPP , http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_82/Docs/SP-181120.zip.
- [SBA-ENHANCEMENT] 3gpp-sba-enhancement, "S2-182904, New SID for Enhancements to the Service-Based 5G System Architecture.", 3GPP , February 2018 (http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip).
- [SDN-DEFINITION] Open Networking Foundation, available at <https://www.opennetworking.org/sdn-definition/>, "Software-Defined Networking (SDN) Definition", 2018.
- [TS23.501] 3gpp-23.501, "Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2 (Rel.15)", 3GPP , December 2018.
- [TS23.502] 3gpp-23.502, "Technical Specification Group Services and System Aspects; Procedures for the 5G System; Stage 2 (Rel. 15)", 3GPP , January 2019.
- [TS29.500] 3gpp-29.500, "Technical Realization of Service Based Architecture.", 3GPP , January 2018.

Authors' Addresses

Dirk Trossen
InterDigital Inc.
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com
URI: <http://www.InterDigital.com/>

Chonggang Wang
InterDigital Inc.
1001 E Hector St, Suite 300
Conshohocken PA 19428
United States

Email: Chonggang.Wang@InterDigital.com
URI: <http://www.InterDigital.com/>

Sebastian Robitzsch
InterDigital Inc.
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Sebastian.Robitzsch@InterDigital.com
URI: <http://www.InterDigital.com/>

Martin Reed
Essex University
Colchester
United Kingdom

Email: mjreed@essec.ac.uk
URI: <https://www.essex.ac.uk/people/reedm58703/martin-reed>

Mays Al-Naday
Essex University
Colchester
United Kingdom

Email: mfhaln@essec.ac.uk
URI: <https://www.essex.ac.uk/people/alned81405/mays-al-naday>

Janne Riihijarvi
RWTH Aachen
Aachen
Germany

Email: jariihij@gmail.com

URI: <https://www.inets.rwth-aachen.de/about-us/janne-riihijaervi/>

ICNRG
Internet-Draft
Intended status: Informational
Expires: April 4, 2021

D. Trossen
Huawei
S. Robitzsch
InterDigital Inc.
M. Reed
M. Al-Naday
Essex University
J. Riihijarvi
RWTH Aachen
October 1, 2020

Internet Services over ICN in 5G LAN Environments
draft-trossen-icnrg-internet-icn-5glan-04

Abstract

In this draft, we provide architecture and operations for enabling Internet services over ICN over (5G-enabled) LAN environments. Operations include ICN API to upper layers, HTTP over ICN, Service Proxy Operations, ICN Flow Management, Name Resolution, Mobility Handling, and Dual Stack Device Support.

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 <https://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 April 4, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Use Cases	5
3.1. 5G Control Plane Services	5
3.2. HTTP-based Streaming	6
4. 5GLAN in 5G Next Generation Core Network Architecture	7
4.1. Realization in SDN Transport Networks	10
4.2. Realization in Other Transport Networks	10
5. Internet Services over ICN over 5GLAN	11
5.1. General Operations	13
5.2. ICN API to Upper Layers	14
5.3. HTTP over ICN	15
5.3.1. General Mapping Procedures	15
5.3.2. Realizing Ad-Hoc Multicast Responses for HTTP	15
5.4. IP over ICN	16
5.5. Service Proxy Operations	17
5.6. Support for Transport Layer Security	17
5.7. ICN Flow Management	18
5.8. NR Operations	21
5.9. Mobility Handling	21
5.10. Dual Stack Device Support	21
6. Deployment Considerations	21
7. Conclusion	22
8. IANA Considerations	23
9. Security Considerations	23
10. Acknowledgments	23
11. Informative References	23
Authors' Addresses	26

1. Introduction

As discussed in [I-D.irtf-icnrg-5gc-icn], Information-Centric Networks (ICN) could be more easily implemented in a Local Area Network (LAN) environment. In relation to 5G, this specifically would realize an ICN deployment without requiring integration of ICN capabilities into the 5G core network itself.

In the currently defined 5G core network, 5GLAN capabilities are being introduced that provide a LAN abstraction to 5G endpoints,

allowing for Ethernet packets to be sent across a 5G network, therefore extending the provisioning of LAN capabilities from fixed and Wifi-based networks to cellular ones.

Utilizing such ICN realization over 5GLAN, the objective of this draft is to propose an architecture to enable Internet services over such ICN-over-LAN environment with the reference architectural discussions in the 5G core network 3GPP specifications [TS23.501] [TS23.502] forming the basis of our discussions. This draft also complements work related to various ICN deployment opportunities explored in [RFC8763], where 5G technology is considered as one of the promising alternatives. In that, ICN is used as an underlay technology to provide routing capabilities to Internet services.

Through such replacement of IP routing with ICN routing, we capitalize on several ICN capabilities:

- o Edge Computing: Multi-access Edge Computing (MEC) is located at the edge of the network and aids several latency sensitive applications such as augmented and virtual reality (AR/VR), as well as the ultra reliable and low latency class (URLLC) of applications such as autonomous vehicles. Enabling edge computing over an IP converged 5GC comes with the challenge of application level reconfiguration required to re-initialize a session whenever it is being served by a non-optimal service instance topologically. In contrast, named-based networking, as considered by ICN, naturally supports service-centric networking, which minimizes network related configuration for applications and allows fast resolution for named service instances. This opportunity is realized by interpreting Internet services as transactions over an ICN routed network with flexible routing to the nearest execution point for said transaction.
- o Edge Storage and Caching: A principal design feature of ICN is the secured content (or named data) object, which allows location independent data replication at strategic storage points in the network, or data dissemination through ICN routers by means of opportunistic caching. These features benefit both real-time and non-real-time applications whenever there is spatial and temporal correlation among content accessed by these users, thereby advantageous to both high-bandwidth and low-latency applications such as conferencing, AR/VR, and non-real time applications such as Video-on-Demand (VOD) and IoT transactions. This opportunity is realized by the transaction-based model of realizing Internet service on top of an ICN routed network, where transaction results can be retrieved from a number of network locations.

- o **Opportunistic Multicast:** The vast majority of current Internet traffic is due to unicast delivery of relatively immutable content such as video or software to very large client groups. This has resulted in large amount of redundancy in network traffic, as well as creating capacity bottlenecks both in the core network as well as the server infrastructure serving the content. Technologies such as content delivery networks (CDNs) help to spread out the network load, but are complex to manage, have inherent limits in terms of how rapidly they can react to changing network and server conditions, and cannot fundamentally reduce the network overhead arising from redundant unicast streams. Furthermore, CDNs traditionally only reach into Points-of-Presence (POP) within customer networks, therefore not reducing the load of transfer from said POP to the end customers in that edge network. In contrast, ICN enables opportunistic multicast delivery of content. We realize this opportunity by automatically delivering responses to quasi-concurrent requests in a single lightweight multicast transmission over the L2 customer network, extending the reach of CDNs down to the end user. Unlike traditional IP multicast, no setup time overhead is added and no per-flow state is required in the network.

In this document, we first outline possible use cases, capitalizing on the aforementioned ICN capabilities before discussing the proposed extensions to 5G to support a cellular-based LAN connectivity before outlining our proposal to support Internet services over an ICN-routed LAN connectivity in such 5G environments.

2. Terminology

Following are terminologies relevant to this draft:

5G-NextGen Core (5GC): Refers to the new 5G core network architecture being developed by 3GPP, we specifically refer to the architectural discussions in [TS23.501] [TS23.502].

5GLAN: Refers to the extensions to the new 5G core network architecture that provide LAN connectivity to 5G devices connected via, e.g., new 5G air interfaces.

User Plane Function (UPF): UPF is the generalized logical data plane function with context of the UE PDU session. UPFs can play many roles, such as, being a flow classifier, a PDU session anchoring point, or a branching point.

Packet Data Network (PDN or DN): This refers to service networks that belong to the operator or third party offered as a service to the UE.

Unified Data Management (UDM): Realizes unified data management for wireless, wireline and any other types of subscribers for M2M, IOT applications, etc. UDM reports subscriber related vital information e.g. virtual edge region, list of location visits, sessions active etc. UDM works as a subscriber anchor point so that means OSS/BSS systems will have centralized monitoring-of/access-to of the system to get/set subscriber information.

Authentication Server Function (AUSF): Provides mechanism for unified authentication for subscribers related to wireless, wireline and any other types of subscribers such as M2M and IOT applications. The functions performed by AUSF are similar to HSS with additional functionalities to related to 5G.

Session Management Function (SMF): Performs session management functions for attached users equipment (UE) in the 5G Core. SMF can thus be formed by leveraging the Control and User Plane Separation (CUPS) feature with control plane session management.

Access Mobility Function (AMF): Perform access mobility management for attached user equipment (UE) to the 5G core network. The function includes, network access stratus (NAS) mobility functions such as authentication and authorization.

Application Function (AF): Helps with influencing the user plane routing state in 5GC considering service requirements.

Network Slicing: This conceptualizes the grouping for a set of logical or physical network functions with its own or shared control, data and service plane to meet specific service requirements.

3. Use Cases

3.1. 5G Control Plane Services

We exemplify the need for chaining service functions at the level of a service name through a use case stemming from the current 3GPP Rel-16 work on Service Based Architecture (SBA) [TS29.500] [SBA-ENHANCEMENT]. In this work, mobile network control planes are proposed to be realized by replacing the traditional network function interfaces with a fully service-based one. HTTP/2 was chosen as the application layer protocol for exchanging suitable service requests [TS29.500]. With this in mind, the exchange between, say the 3GPP (Rel-15) defined Session Management Function (SMF) and the Access and Mobility management Function (AMF) in a 5G control plane is being described as a set of web service like requests which are in turn embedded into HTTP requests. Hence, interactions in a 5G control

plane can be modelled based on service function chains where the relationship is between the specific service function endpoints that implement the necessary service endpoints in the SMF and AMF. The service functions are exposed through URIs with work ongoing to define the used naming conventions for such URIs.

This move from a network function model (in pre-Rel 15 systems of 3GPP) to a service-based model is motivated through the proliferation of data center operations for mobile network control plane services. In other words, typical IT-based methods to service provisioning, in particular that of virtualization of entire compute resources, are envisioned to being used in future operations of mobile networks. Hence, operators of such future mobile networks desire to virtualize service function endpoints and direct (control plane) traffic to the most appropriate current service instance in the most appropriate (local) data centre, such data centre envisioned as being interconnected through a software-defined wide area network (SD-WAN). 'Appropriate' here can be defined by topological or geographical proximity of the service initiator to the service function endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data center centric operation is extended with the trend towards regionalization of load through a 'regional office' approach, where micro data centers provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the 'most appropriate' service endpoint for a particular incoming service request. This 5G control plane scenario capitalizes on the edge computing capabilities of ICN by allowing for fast redirections of HTTP-based transactions to the nearest control plane service realization within the distributed data centre of the 5G operator infrastructure.

3.2. HTTP-based Streaming

With the extensive use of "web technology", "distributed services" and availability of heterogeneous network, HTTP has effectively transitioned into the common transport or session layer for E2E and multi-hop communication across the web. Assume clients that are consuming the same content (such as a TV program) and that this content has for each block (typically segments worth 2 seconds of content) a set of outstanding requests from its clients. HTTP request and response used in media streaming services like HLS, use HTTP response for delivery of content. In such scenarios, where semi-synchronous access to the same resource occurs (such as watching prominent videos over Netflix or similar platforms or live TV over HTTP), traffic grows linearly with the number of viewers since the HTTP-based server will provide an HTTP response to each individual

viewer. To mitigate the load impact, operators often utilize IP multicast underneath HTTP (for live TV) to create fewer, multicast, streams; though this comes with the high flow setup and management cost. This poses a significant burden on operators in terms of costs and on users in terms of likely degradation of quality.

This problem is not limited to traditional TV broadcasting. Consider a virtual reality use case where several users are joining a VR session at the same time, e.g., centered around a joint event. Hence, due to the temporal correlation of the VR sessions, we can assume that multiple requests are sent for the same content at any point, particularly when viewing angles of VR clients are similar or the same. Due to availability of virtual functions and cloud technology, the actual end point from where content is delivered may change. For this type of scenarios, the opportunistic multicast capability of ICN may be utilized to reduce overall load in the network, as well as on the server providing the HTTP responses. The latter also allows constrained resources to serve a higher volume of demands and therefore incur a higher impact on traffic distribution in the network.

4. 5GLAN in 5G Next Generation Core Network Architecture

In this section, for brevity purposes, we restrict the discussions to the 5G extensions currently studied in 3GPP to facilitate a distributed, cellular-based LAN connectivity to end users, based on the 5G next generation core network architecture. For more information on the latter, we refer to [TS23.501] [TS23.502] as well as [I-D.irtf-icnrg-5gc-icn].

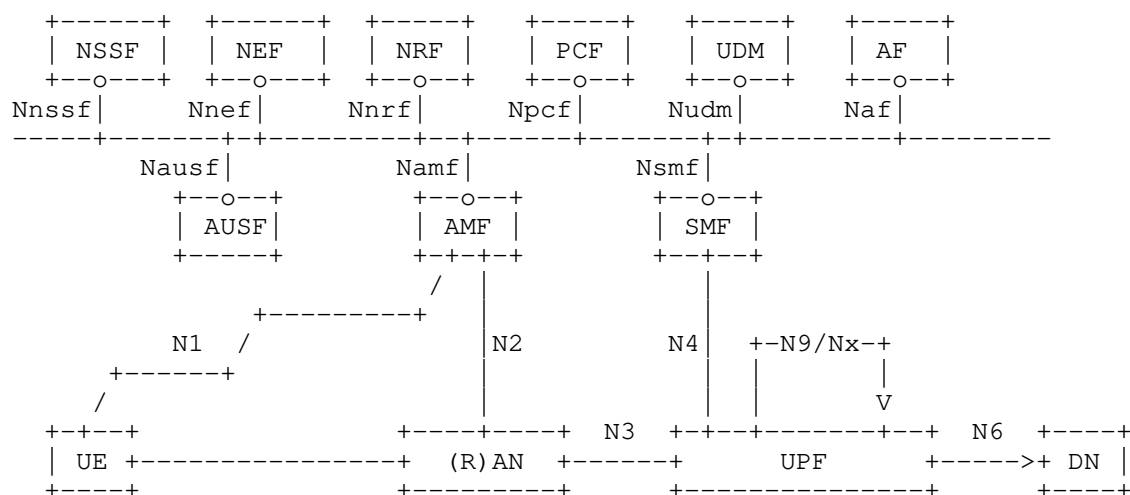


Figure 1: 5G Core Network with Vertical LAN (5GLAN) Extensions

Figure 1 shows the current 5G Core Network Architecture being discussed within the scope of the normative work addressing 5GLAN Type services in the 3GPP System Architecture Working Group 2 (3GPP SA2), referred formally as "5GS Enhanced support of Vertical and LAN Services" [SA2-5GLAN]. The goal of this work item is to provide distributed LAN-based connectivity between two or more terminals or User Equipment entities (UEs) connected to the 5G network. The SMF (session management function) provides a registration and discovery protocol that allows UEs wanting to communicate via a relevant 5GLAN group towards one or more UEs also members of this 5GLAN group, to determine the suitable forwarding information after each UE previously registered suitable identifier information with the SMF responsible to manage the paths across UEs in a 5GLAN group. UEs register and discover (obtain) suitable identifiers during the establishment of a Protocol Data Unit (PDU) Session or PDU Session Modification procedure. Suitable identifier information, according to [SA2-5GLAN], are Ethernet MAC addresses as well as IP addresses (the latter is usually assigned during the session setup through the SMF, i.e., the session management function).

The SMF that manages the path across UEs in a 5GLAN group, then establishes the suitable procedures to ensure the forwarding between the required UPFs (user plane functions) to ensure the LAN connectivity between the UEs (user equipments) provided in the original request to the SMF. When using the N9 interface to the UPF, this forwarding will rely on a tunnel-based approach between the UPFs along the path, while the Nx interface uses path-based forwarding

between UPFs, while LAN-based forwarding is utilized between the final UPF and the UE (utilizing the N3 interface towards the destination UE).

In the following, path-based forwarding is assumed, i.e., the usage of the Nx interface and the utilization of a path identifier for the end-to-end LAN communication. Here, the path between the source and destination UPFs is encoded through a bitfield, provided in the packet header. Each bit position in said bitfield represents a unique link in the network. Upon receiving an incoming packet, each UPF inspects said bitfield for the presence of any local link that is being served by one of its output ports. Such presence check is implemented via a simple binary AND and CMP operation. If no link is being found, the packet is dropped. Such bitfield-based path representation also allows for creating multicast relations in an ad hoc manner by combining two or more path identifiers through a binary OR operation. Note that due to the assignment of a bit position to a link, path identifiers are bidirectional and can therefore be used for request/response communication without incurring any need for path computation on the return path.

For sending a packet from one Layer 2 device (UE) connected to one UPF (via a RAN) to a device connected to another UPF, we provide the MAC address of the destination and perform a header re-write by providing the destination MAC address of the ingress UPF when sending from source device to ingress and placing the end destination MAC address in the payload. Upon arrival at the egress UPF, after having applied the path-based forwarding between ingress and egress UPF, the end destination address is restored while the end source MAC is placed in the payload with the egress L2 forwarder one being used as the L2 source MAC for the link-local transfer. At the end device (or proxy device), the end source MAC address is restored as the source MAC, providing an abstraction of a link-local L2 communication between the end source and destination devices.

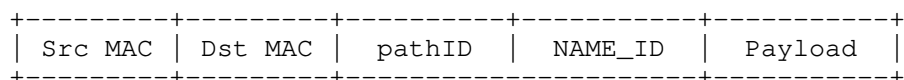


Figure 2: General Packet Structure

For this end-to-end transfer, the general packet structure of Figure 2 is used. The Name_ID field is being used for the ICN operations, while the payload contains the information related to the transaction-based flow management described in Section 5.8 and the

PATH_ID is the bitfield-based path identifier for the path-based forwarding.

4.1. Realization in SDN Transport Networks

An emerging technology for Layer 2 forwarding that suits the 5GLAN architecture in Figure 1 is that of Software-Defined networking (SDN) [SDN-DEFINITION], which allows for programmatically forwarding packets at Layer 2. Switch-based rules are being executed with such rules being populated by the SDN controller. Rules can act upon so-called matching fields, as defined by the OpenFlow protocol specification [OpenFlowSwitch]. Those fields include Ethernet MAC addresses, IPv4/6 source and destination addresses and other well-known Layer 3 and even 4 transport fields.

As shown in [Reed], efficient path-based forwarding can be realized in SDN networks by placing the aforementioned path identifiers into the IPv6 source/destination fields of a forwarded packet. Utilizing the IPv6 source/destination fields allows for natively supporting 256 links in a transport network. Larger topologies can be supported by extension schemes but are left out of this paper for brevity of the presentation. During network bootstrapping, each link at each switch is assigned a unique bitnumber in the bitfield. In order to forward based on such bitfield path information, the SDN controller is instructed to insert a suitable wildcard matching rule into the SDN switch. This wildcard at a given switch is defined by the bitnumber that has been assigned to a particular link at that switch during bootstrapping. Wildcard matching as a generalization of longest prefix matching is natively supported by SDN-based switches since the OpenFlow v1.3 specification, efficiently implemented through TCAM based operations. With that, SDN forwarding actions only depend on the switch-local number of output ports, while being able to transport any number of higher-layer flows over the same transport network without specific flow rules being necessary. This results in a constant forwarding table size while no controller-switch interaction is necessary for any flow setup; only changes in forwarding topology (resulting in a change of port to bit number assignment) will require suitable changes of forwarding rules in switches.

4.2. Realization in Other Transport Networks

Although we focus the methods in this draft on Layer 2 forwarding approaches and realization of Internet-over-ICN over a 5G LAN enabled network, path-based transport networks can also be established as an overlay over otherwise Layer 2 networks. For instance, the BIER (Bit Indexed Explicit Replication) [RFC8279] efforts within the Internet Engineer Task Force (IETF) establish such path-based forwarding

transport as an overlay over existing, e.g., MPLS networks. The path-based forwarding identification is similar to the aforementioned SDN realization although the bitfield represents ingress/egress information rather than links along the path.

Yet another transport network example is presented in [Khalili], utilizing flow aggregation over SDN networks. The flow aggregation again results in a path representation that is independent from the specific flows traversing the network.

The proposed traffic engineering extensions to BIER, presented in [I-D.ietf-bier-te-arch], directly align with the SDN-based realization presented in Section 4.1, by proposing the same bitposition per transport link assignment being used, resulting in BIER bitstrings in which a dedicated forwarding path is encoded as a unique bitpattern containing said bitpositions of the chosen forwarding links. The BIER-TE controller plays a similar role as the northbound SDN controller application utilized for the solution in Section 4.1.

5. Internet Services over ICN over 5GLAN

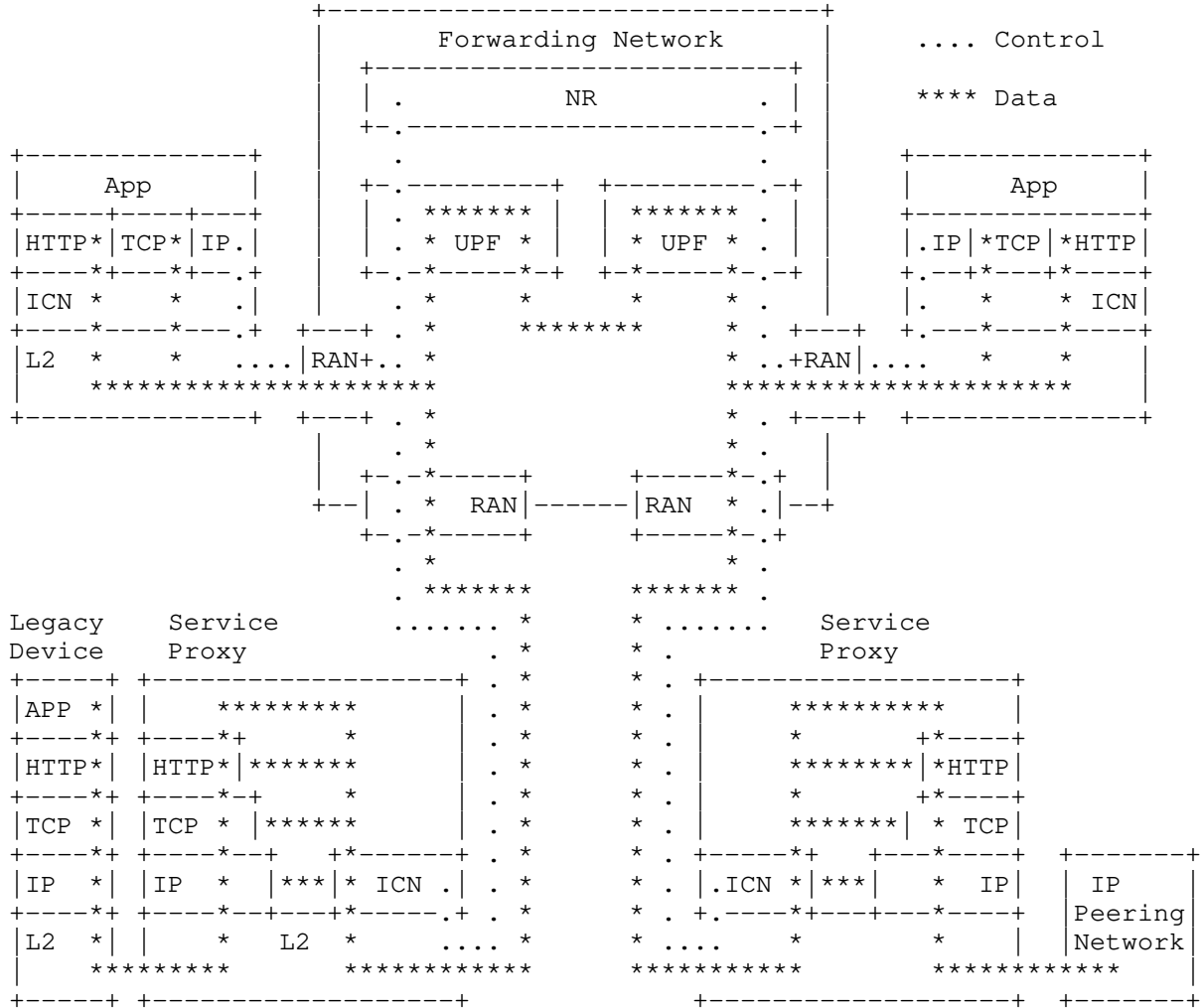


Figure 3: Internet Services over ICN over 5GLAN

Figure 3 shows the protocol layering for realizing Internet protocols over an ICN over 5GLAN transport, assuming an end-to-end LAN connectivity provided by solutions such as 5GLAN.

Note that such LAN connectivity can also be found in environments such as localized LAN-based deployments in smart cities, enterprises and others, with the UPF representing, e.g., an SDN switch (utilizing the methods outlined in Section 4.1). Hence, the solutions described in this section also applies to those deployments.

Key to the approach is that Internet services are being interpreted as the main unit of transfer in the architecture shown in Figure 3. For this, any Internet service is treated as a Named Service Transaction (NST) which is in turn suitably routed over an ICN layer in one or more other devices. As a result of this name-based interpretation of any Internet service, the protocol stack in end devices flattens to four layers with Internet services and ICN, with ICN acting as a name-based routing layer for all IP protocols implemented atop, with Layer 1 and 2 realizing the end-to-end packet forwarding outlined in Section 4 (over a 5G environment) or a general LAN environment provided through WiFi or fixed Ethernet technologies.

The general ICN operations are presented in Section 5.1 before discussing the assumed (strawman) API to the ICN layer in Section 5.2, which is used in turn to define the mapping of HTTP transactions to operations at the ICN layer in Section 5.3 for the example of HTTP. As explained in that section, the ICN layer uses an interaction with the NR to register and discover HTTP-based services for determining the suitable end-to-end packet forwarding information.

Interfaces to legacy devices and peering networks are preserved through service proxy devices, which terminate a traditional Internet protocol stack communication and translate it into a resulting flat protocol transaction. Termination here can be based on well-known port numbers for specific Internet protocols, ultimately falling back to the IP datagram service being the minimal service being mapped. The operations of said service proxy devices is described in Section 5.5.

An important aspect of the architecture is the mapping of the end-to-end flow semantic established in many Internet services onto the flat protocol stack. Section 5.7 outlines the flow management that exists between the end devices.

The mapping of protocol identifiers onto ICN forwarding relations, i.e., the operations of the name resolver (NR), shown in Figure 3, is described in Section 5.8, followed by the procedures for handling mobility of service providers and consumers in Section 5.9. Finally, the support for dual-stack devices, not requiring a service proxy device yet being able to also connect to existing IP routed networks, is described in Section 5.10.

5.1. General Operations

The semantics of our name-based routing is that of a publish-subscribe system over a name. The intention to receive packets with a certain name is expressed through a subscription while sending

packets to a name is expressed through a publication. The matching of a sender to a receiver is realized through NR in Figure 3. The exact nature of the matching is defined through the semantics of the service and, therefore, through the nature of the name provided. For instance, HTTP and raw Internet services are matched to exactly one subscriber only, providing an anycast capability, while IP multicast services are matched against any subscriber (with the IP multicast address being the name).

Structured names are used with the root specific to the (Internet) service name, such as URL, and therefore deriving the matching semantics directly from the name.

The subscription to a name is realized through a registration protocol between end device and NR. Hence, any end device exposing a certain Internet service registers the suitable name with the NR, which in turn stores reachability information that is suitable for path calculation between the ingress and egress L2 forwarders between which the communication eventually will take place. In our current realization, we utilize shortest paths only although other link weights can be utilized for, e.g., delay-constrained and other policies.

In our realization, we use network domain unique host identifiers that are being assigned to end devices during the connectivity setup. Sending a packet of a given Internet service is realized through a discovery protocol, which returns a suitable pathID, i.e., the forwarding information between ingress and egress L2 forwarder, and the destination MAC address of the hosting end device. It is this pathID and MAC address that is being used in the general packet structure of Figure 2 to forward the packet to the destination.

To reduce latency in further communication, the forwarding information is locally cached at the end device, while the cached information is being maintained through path updates sent by the NR in case of hosting end devices having moved or de-registered, therefore avoiding stale forwarding information.

5.2. ICN API to Upper Layers

The operations of the ICN layer are exposed to upper layers in Figure 1 through the following API calls, being exemplary here for the further explanation of operations in the next sub-sections:

- o conn = send(name, payload)
- o send(conn, payload)

```
o conn = receive(name, &payload)

o receive(conn, &payload)
```

The first `send()` call is used for initiating a send operation to a name with a connection handle returned, while the second `send()` is used for return calls, using a connection parameters that is being received with the `receive()` call to an incoming connection or for subsequence outgoing calls after an initial request to a name has been made. A return `send()` is being received at the other (client) side through the second `receive()` call where the `conn` parameter is obtained by the corresponding `send()` call for the outgoing call. With these API functions, we provide means for providing name-based transactions with return responses association provided natively.

The `conn` parameter represents the bitfield used for path-based forwarding in the remote host case or the hash of the local MAC address in case of link-local connections.

5.3. HTTP over ICN

5.3.1. General Mapping Procedures

To realize the flat device nature, Internet service layers, such as the HTTP protocol stack or the TCP protocol stack, will need to be adapted to run atop this new API, implementing the semantics of the respective Internet protocol through suitable transactions at the name level. In the example of HTTP, the standard operations of DNS resolution for the server to be contacted and opening of a TCP (for HTTP/1.1 or HTTP/2) or UDP (for HTTP/3) socket are altogether replaced by a single `send(FQDN, HTTP request)` call; but the response will be sent by the server, which received the request through a `receive(FQDN, &payload)` call, using the returned `conn` parameter to send the response with the second `send()` API call. Note that the use of bidirectional pathIDs, no NR lookup is performed at the HTTP serving endpoint.

In the light of HTTP/3, the same mappings apply as already described above with the exception that the service proxy intercepts incoming UDP traffic only if it carries an HTTP/3 payload. If the payload is not HTTP/3, the mappings as described in Section Section 5.4 apply.

5.3.2. Realizing Ad-Hoc Multicast Responses for HTTP

The basis of a named service transaction allows to deliver the same HTTP responses to several requestees in efficient multicast (see [I-D.ietf-bier-multicast-http-response] for use cases in a BIER-based transport network environment).

This opportunity is realized by sending the same payload (i.e., an HTTP response to the same resource across a number of pending requests) through a combination of several conn parameters received in the incoming requests via the receive() function.

What is required in the HTTP stack implementation is a logic to decide that two or more outstanding requests are possible to be served by one response. For this, upon receiving an incoming request, the HTTP stack determines any outstanding request to the same resource. 'Same' here is defined as URI-specific combination of the request URI and URI-specific header fields, such as browsing agent or similar, called requestID in the following.

Once such determination is made that two requests are relating to the same resource, i.e., are having the same request ID, the HTTP stack maintains a temporary mapping of the request ID to the respective conn parameters delivered by the receive() call. Upon receiving the HTTP response from its application-level logic, the HTTP stack will generate the suitable send(conn, payload) call where the provided conn parameter is bitwise OR of all previously stored conn parameters received in the receive() call. The ICN layer will recognize the use of those ad-hoc created conn parameters and set the destination MAC address in the general packet structure of Figure 2 to the Ethernet broadcast MAC address as the destination address, leading to sending the response to all end devices at the egress L2 forwarders to which the response will be forwarded based on the combined conn parameter. Alternatively, one could request IEEE assignment for a specific Ethernet multicast address for this scheme instead of using the broadcast address. For the local end device to determine the relevance of the response received at the broadcast channel, the HTTP stack of the serving endpoint includes the aforementioned requestID into the payload of the packet (see Figure 2), while the originating endpoint maintains an internal table with the requestID of pending requests and its associated conn handle. If no matching requestID is found, the packet is not being delivered to the ICN layer of the incoming device. If a request is found, the ICN layer delivers the response via the receive() call, using the conn handle stored in the pending request table. Note that this filtering mechanism can easily be implemented in hardware upon standardizing the appropriate payload and header fields.

5.4. IP over ICN

For non-HTTP traffic, the service proxy uses the destination IP address of an incoming IP packet from an IP endpoint as the information identifier for the NR to find the suitable service proxy, which can reach the sought after IP endpoint. The usage of subnet masks and a longest prefix matching approach inside the NR is

foreseen for this matching process. In a 5GLAN scenario, service proxies on UEs serve a single IP endpoint (i.e. the UE itself) and therefore are represented by a /32 subnet mask for its IP address inside the NR in the list of subscribers. Service proxies that serve an entire local domain the subnet configured on the LAN interface of the service proxy is being communicated to the NR for matching purposes. The service proxy that serves the public internet is represented as a wildcard match for any IP address that is not served within the operator's network.

5.5. Service Proxy Operations

The service proxy in Figure 3 serves the integration of legacy devices, i.e., with regular IP protocol stack, and the interconnection to IP-based peering networks. It registers suitable identifiers with the NR to ensure the reception of (ICN) packets, while providing suitable protocol termination for the various supported protocols. For instance, for HTTP, the service proxy towards the peering network will register a wildcard name to the NR to receive any HTTP request not destined to a network-locally registered FQDN, operating as an HTTP proxy towards the peering network. Service proxies also register to the IP subnet they have been assigned on their local IP-based interface(s). The assignment is envisaged through an out-of-band address assignment scheme, i.e. DHCP [RFC2131] [RFC8415].

5.6. Support for Transport Layer Security

With a vast amount of networking intense HTTP traffic being encrypted, supporting HTTP over Transport Layer Security (TLS) [RFC8446] is of paramount importance to continue offering the benefits of routing the internet over 5GLAN utilising the ICN principles outlined in this document, in particular the ability to transparently change the service endpoint handling a HTTP request per HTTP transaction.

After the TCP session has been established by the client with the server, which is transparently intercepted by the service proxy and mapped to an inter service proxy ICN flow, the client initiates a TLS handshake aiming to establish a secure connection. When the service proxy receives the ClientHello message from the client it has two choices: act as a TLS endpoint or act as a TLS proxy.

If the service proxy has the TLS certificate/key for the FQDN provided in the TLS ClientHello message, it acts as a transparent TLS endpoint by intercepting the TLS sessions and implementing the TLS procedures described in [RFC8446]. If the client eventually sends a HTTP request over the TLS session the service proxy can decrypt it to

obtain the HTTP request in its entirety to perform the steps described above how to map HTTP over ICN (and vice versa). On the other side of the ICN flow the service proxy acts as a TLS client towards the actual IP service endpoint. One of the key benefits of service proxies acting as TLS endpoints is their ability to still offer opportunistic multicast, as TLS (similar to TCP) is fully intercepted at both edges of the ICN communication domain.

If the TLS certificate/key for the FQDN in the TLS ClientHello message is not available to the service proxy, TLS control messages between client and servers are left intact and routed to the most suitable service proxy that is subscribed to the FQDN. However, as the service proxy is not able to see HTTP transactions routing benefits of the described steps such as opportunistic multicast and transparent interruption-free re-routing of HTTP transactions to a more suitable IP service endpoint are not feasible. In such scenario, the TLS session must be reestablished between the client and the server and service continuity cannot be offered. However, it is important to mention that the TLS proxy mode still improves over a plain TCP connection, as for the latter the IP address provided by a DNS is being used to determine the destined service proxy and not the FQDN of the HTTP request.

5.7. ICN Flow Management

For all protocol mappings described in this section, the payload taken from the (intercepted) layer is sent as payload of the ICN packet, as illustrated in Figure 2. It can be observed that two resource management regimes are present, i.e. the application to service proxy communication (IP) and the inter service proxy (ICN) one. In the IP resource management regime, TCP friendliness governs the various transport protocols in use allowing a per flow fair usage of the available networking resources. However, the resource regime between service proxies does not have such requirement; thus, the corresponding ICN flow management and error control allows an independent improved resource regime that must not be TCP friendly.

For an independent inter service proxy resource management scheme that treats each *-over-ICN mapping equally, the notion of HTTP, TCP and IP transactions are being introduced with the goal to treat them equally and therefore ensuring resource fairness among them with the benefit of long lasting ICN flow relationships among service proxies.

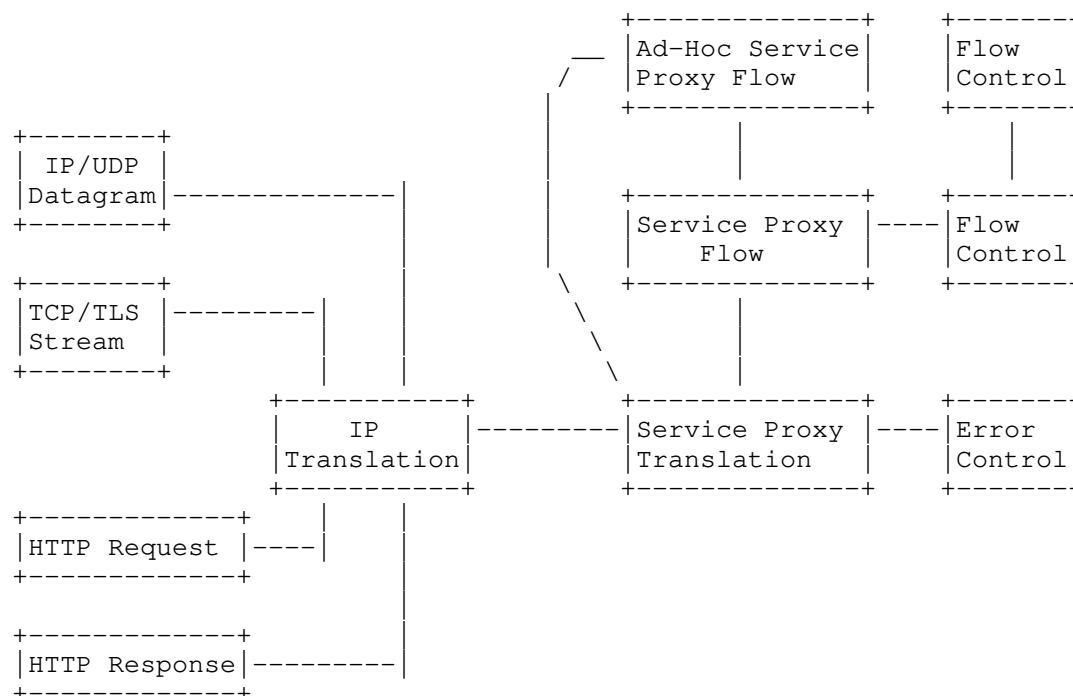


Figure 4: Mapping of IP Transactions onto Service Proxy Transactions and Flows

Figure 4 illustrates the ICN flow management for inter service proxy communication. As it shows, all traffic from IP endpoints are translated into a unified IP transaction and mapped to a service proxy transaction. The resulting service proxy flow constitutes a long-term relationship between two service proxies. For each service proxy flow, there exists a flow-specific flow control relationship, which maintains flow parameters such as send credits, timers for round-trip time (RTT) dependent mechanisms, error rate information and others. Such service proxy flow between two service proxies represent the edge-to-edge resource management regime described above. Each service proxy flow consists of one or more service proxy transactions, each of which comes with its own error control relationship that maintains information such as sequence numbers, outstanding packets, segmentation/reassembly information and others. For retransmissions, the error control relies on service proxy flow-specific flow control information, such as timers, RTT information etc. With such mapping from IP transactions onto a service proxy transaction that has its own error control mechanism, it has been achieved that the data originating from and destined to end-to-edge

resource management regimes can be reliably transferred over the service proxy-to-service proxy network. Combining all transactions under a single resource management relationship, represented by the combined flow control mechanism for a single flow between the service proxies, now establishes the inter service proxy resource management scheme. Any competition for resources among service proxies is now governed by said scheme between flows. Given that all transactions between specific service proxies are mapped into a single service proxy flow, fair resource sharing among all transactions can be ensured.

One crucial aspect of the HTTP-over-ICN mapping is the possibility of so-called ad-hoc multicast relations, i.e., the ability to send responses from one IP applications to more than one other IP application and therefore to more than one service proxy. In this case, the specific IP transaction (e.g. an HTTP response) is mapped onto a service proxy transaction that in turn is realized over more than one service proxy-to-service proxy flow. This flow is called ad-hoc service proxy flow. For those cases, the flow control for the ad-hoc service proxy flow will utilize parameters across the various involved service proxy flows, resulting in an one-to-many relationship between the specific flow control for the ad-hoc service proxy flow and the flow control(s) of the involved service proxy flow(s). Such combined parameters might be the maximum RTT timer or the lowest credit value, representing the least common dominator of the resources across all involved flows.

As mentioned before, a service proxy flow constitutes a long-term relationship between two service proxies. This relationship can be established in multiple ways: an explicit setup might be used akin to that of TCP's three-way handshake. Alternatively, an implicit transaction-based flow establishment might be used; in this case, the sending of an initial transaction between two service proxies results in the creation of an service proxy flow context between those two service proxies, which is being reused for any future transfer between those two service proxies, i.e., constituting a service proxy flow. Flow termination can be explicit based on a handshake protocol, where one service proxy, wishing to terminate the flow, signals this to the corresponding service proxy. Other embodiments foresee the destruction of the service proxy flow via timeout, e.g., removing any internal service proxy flow context information upon firing of an inactivity timeout. Combining this with an implicit transaction-based flow establishment would make the notion of a service proxy flow entirely that of an internal (service proxy flow context) data structure, which is created upon sending the first transaction to a service proxy which had previously not been contacted, while destroying said data structure upon the firing of the aforementioned inactivity timeout.

5.8. NR Operations

The NR in Figure 3 combines the operations of the SMF and the PMF in 5GLAN (see Figure 1), by allowing for registering IP protocol identifiers for discovery and subsequent path computation by resolving the destination(s) to a suitable pathID and destination MAC address for forwarding. This will require extensions to the operations of the SMF to allow for such higher layer identifiers to be registered (and discovered), in addition to the already supported Ethernet and IP addresses.

5.9. Mobility Handling

EDITOR NOTE: left for future draft updates.

5.10. Dual Stack Device Support

Figure 3 outlines a protocol stack for the user equipment that realizes Internet services on top of the proposed name-based routing layer as a single stack device. However, [I-D.irtf-icnrg-icn-lte-4g] outlines the possibility of supporting dual-stack devices for 4G LTE networks by allowing IP as well as ICN protocol stacks to be deployed with the operation of IP and ICN based applications. [I-D.irtf-icnrg-5gc-icn] outlines the same dual-stack device realization for a 5G ICN realization. For both environments, a convergence layer is described that selects the appropriate data path for each ICN or IP application, e.g., based on configuration per application (similar to selecting network interfaces such as WiFi over cellular).

As a possible data path selection, [I-D.irtf-icnrg-icn-lte-4g] and [I-D.irtf-icnrg-5gc-icn] envision the realization of Internet-over-ICN (Section 4.2 in [I-D.irtf-icnrg-icn-lte-4g]) in which the convergence layer would realize similar mapping functions as described in this draft. Hence, we foresee the utilization of such dual-stack devices connected to an Internet services over ICN over 5GLAN environment. When utilizing the service proxy, IP applications that are configured to use the IP data path only could still utilize the ICN-based forwarding in the network. In that case, functionality such as the opportunistic multicast in Section 5.3.2 would only reach up to the service proxy with unicast traffic continuing along the data path towards the user equipment.

6. Deployment Considerations

The work in [RFC8763] outlines a comprehensive set of considerations related to the deployment of ICN. We now relate the solutions proposed in this draft to the two main aspects covered in the

deployment considerations draft, namely the 'deployment configuration' (covered in Section 3 of [RFC8763]) that is being realized and the 'deployment migration paths' (covered in Section 4 of [RFC8763]) that are being provided.

The solutions proposed in this draft relate to those "deployment configuration" as follows:

- o The realization of Internet service on top of an ICN routing capabilities, as proposed in Section 5, follows the "ICN-as-an-Underlay" categorization, interpreting the ICN routing as an underlay to the Internet services with the path-based forwarding being compatible with the 5GLAN forwarding capabilities currently discussed in 3GPP and therefore providing an underlay integration capability for the ICN forwarding used in the proposed solution.
- o The deployment of 5GLAN based ICN capabilities can be realized following the "ICN-as-a-Slice" deployment configuration, i.e., the 5GLAN connectivity is provided to a "vertical 5G customer" which in turn provides the ICN capability over 5GLAN within said network (and compute) slice at the endpoints of the 5GLAN connectivity, as proposed in Section 3.

In relation of the 'deployment migration paths', the solutions in this draft relate as follows:

- o The integration with the 5GLAN capability, as proposed in Section 5, facilitates "edge network migration" (interpreting the cellular sub-system here as an edge network albeit a possibly geographically large one).
- o The single stack realization, as proposed in Figure 3, as well as the dual-stack deployment, as proposed in Section 5.10, facilitate "application and services migration" through not only supporting ICN applications but also Internet applications through the proposed Internet-over-ICN mapping in the terminal.
- o The Internet over ICN over 5GLAN deployment, possibly combined with an ICN-as-a-Slice deployment, facilitates the "content delivery networks migration" through a deployment of Internet-over-ICN-based 5GLAN connected CDN elements in (virtualized) edge network nodes or POP locations in the customer (5G) network.

7. Conclusion

In this draft, we explored the feasibility of enabling Internet services directly over ICN network over (5G)LAN environments. We proposed the architecture and discussed corresponding operations of

mapping Internet services onto name-based transactions, with the specific example of HTTP-based transactions. We described the flow management, the realization of opportunistic multicast responses for HTTP as well as the realization of dual-stack user equipment. Future updates to the draft will provide more details to mobility handling. We also described the deployment scenario for supporting Internet services over ICN over 5GLAN.

8. IANA Considerations

This document requests no IANA actions.

9. Security Considerations

Editor Note: to be added in future drafts.

10. Acknowledgments

Work towards developing the solutions outlined in this draft have been funded under grants of the [H2020POINT] and [H2020FLAME] projects.

11. Informative References

[H2020FLAME]

H2020, "The FLAME Project", <https://www.ict-flame.eu/> .

[H2020POINT]

H2020, "The POINT Project", <https://www.point-h2020.eu/> .

[I-D.galis-anima-autonomic-slice-networking]

Galis, A., Makhijani, K., Yu, D., and B. Liu, "Autonomic Slice Networking", draft-galis-anima-autonomic-slice-networking-05 (work in progress), September 2018.

[I-D.ietf-bier-multicast-http-response]

Trossen, D., Rahman, A., Wang, C., and T. Eckert, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", draft-ietf-bier-multicast-http-response-04 (work in progress), July 2020.

[I-D.ietf-bier-te-arch]

Eckert, T., Cauchie, G., and M. Menth, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", draft-ietf-bier-te-arch-08 (work in progress), July 2020.

- [I-D.irtf-icnrg-5gc-icn]
Ravindran, R., suthar, P., Trossen, D., Wang, C., and G. White, "Enabling ICN in 3GPP's 5G NextGen Core Architecture", draft-irtf-icnrg-5gc-icn-03 (work in progress), July 2020.
- [I-D.irtf-icnrg-icn-lte-4g]
suthar, P., Stolic, M., Jangam, A., Trossen, D., and R. Ravindran, "Native Deployment of ICN in LTE, 4G Mobile Networks", draft-irtf-icnrg-icn-lte-4g-08 (work in progress), July 2020.
- [I-D.muscariello-intarea-hicn]
Muscariello, L., Carofiglio, G., Auge, J., Papalini, M., and M. Sardara, "Hybrid Information-Centric Networking", draft-muscariello-intarea-hicn-04 (work in progress), May 2020.
- [I-D.white-icnrg-ipoc]
White, G., Shannigrahi, S., and C. Fan, "Internet Protocol Tunneling over Content Centric Mobile Networks", draft-white-icnrg-ipoc-02 (work in progress), June 2019.
- [Khalili] Khalili, R., Poe, W., Despotovic, Z., and A. Hecker, "Reducing State of SDN Switches in Mobile Core Networks by Flow Rule Aggregation", IEEE ICCCN 2016, Hawaii, USA, August 2016.
- [OpenFlowSwitch]
Open Networking Foundation, available at <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>, "OpenFlow Switch Specification V1.5.1", 2018.
- [Reed] Reed, M., AI-Naday, M., Thomos, N., Trossen, D., Petropoulos, G., and S. Spirou, "Stateless Multicast Switching in Software Defined Networks", IEEE ICC 2016, Kuala Lumpur, Malaysia, 2016.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlich, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <<https://www.rfc-editor.org/info/rfc7927>>.

- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8763] Rahman, A., Trossen, D., Kutscher, D., and R. Ravindran, "Deployment Considerations for Information-Centric Networking (ICN)", RFC 8763, DOI 10.17487/RFC8763, April 2020, <<https://www.rfc-editor.org/info/rfc8763>>.
- [SA2-5GLAN] 3gpp-5glan, "SP-181129, Work Item Description, Vertical_LAN(SA2), 5GS Enhanced Support of Vertical and LAN Services", 3GPP , http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_82/Docs/SP-181120.zip.
- [SBA-ENHANCEMENT] 3gpp-sba-enhancement, "S2-182904, New SID for Enhancements to the Service-Based 5G System Architecture.", 3GPP , February 2018 (http://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip).
- [SDN-DEFINITION] Open Networking Foundation, available at <https://www.opennetworking.org/sdn-definition/>, "Software-Defined Networking (SDN) Definition", 2018.
- [TS23.501] 3gpp-23.501, "Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2 (Rel.15)", 3GPP , December 2018.
- [TS23.502] 3gpp-23.502, "Technical Specification Group Services and System Aspects; Procedures for the 5G System; Stage 2 (Rel. 15)", 3GPP , January 2019.

[TS29.500]

3gpp-29.500, "Technical Realization of Service Based
Architecture.", 3GPP , January 2018.

Authors' Addresses

Dirk Trossen
Huawei Technologies Duesseldorf GmbH
205 Hansallee
Duesseldorf 40549
Germany

Email: dirk.trossen@huawei.com
URI: <http://huawei-dialog.de/>

Sebastian Robitzsch
InterDigital Inc.
64 Great Eastern Street, 1st Floor
London EC2A 3QR
United Kingdom

Email: Sebastian.Robitzsch@InterDigital.com
URI: <http://www.InterDigital.com/>

Martin Reed
Essex University

Colchester
United Kingdom

Email: mjreed@essex.ac.uk
URI: <https://www.essex.ac.uk/people/reedm58703/martin-reed>

Mays Al-Naday
Essex University

Colchester
United Kingdom

Email: mfhaln@essex.ac.uk
URI: <https://www.essex.ac.uk/people/alned81405/mays-al-naday>

Janne Riihijarvi
RWTH Aachen

Aachen
Germany

Email: jariihi@googlemail.com

URI: <https://www.inets.rwth-aachen.de/about-us/janne-riihijaervi/>