ICN Research Group
Internet-Draft
Intended status: Experimental
Expires: January 17, 2019

C. Gundogan
T. Schmidt
HAW Hamburg
M. Waehlisch
link-lab & FU Berlin
C. Scherb
C. Marxer
C. Tschudin
University of Basel
July 16, 2018

ICN Adaptation to LowPAN Networks (ICN LoWPAN)
draft-gundogan-icnrg-ccnlowpan-02

Abstract

In this document, a convergence layer for CCNx and NDN over IEEE 802.15.4 LoWPAN networks is defined. 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 link fragmentation component of the 6LoWPAN dispatching framework is applied to ICN chunks. Basic improvements in efficiency are advised by stateless and stateful compression schemes.

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 January 17, 2019.

Table of Contents

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 have shown noteable advantages over the traditional host-to-host approach on the Internet [NDN-EXP]. 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.

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 octets. The maximum frame header size is 25 octets, which leaves 102 octets for the payload. IEEE 802.15.4 security features further reduce this payload length by up to 21 octets, yielding a net of 81 octets for CCNx or NDN packet headers, signatures and content.

6LoWPAN [RFC4944][RFC6282] is a convergence layer that provides frame formats, header compression and link 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 IEEE 802.15.4 payload and meta information about link 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 that compresses packet headers of CCNx as well as NDN and allows for an increased payload size per packet. Additionally by reusing the dispatching framwork defined by 6LoWPAN, compatibility between coexisting wireless networks of competing technologies is enabled. This also allows to reuse the link fragmentation scheme specified by 6LoWPAN for ICN LoWPAN.

ICN LoWPAN utilizes 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 largely. For further reductions, default header values suitable for constrained IoT networks are selected in order to elide corresponding TLVs.

In a typical IoT scenario (see Figure 1), embedded devices are interconnected via quasi-stationary infrastructure whith a border router (BR) interconnecting the constrained LoWPAN networks via some Gateway with the public Internet. In ICN based IoT networks, Interest and Data messages transparently travel through the BR up and down between a Gateway and the embedded devices within the constrained LoWPANs.

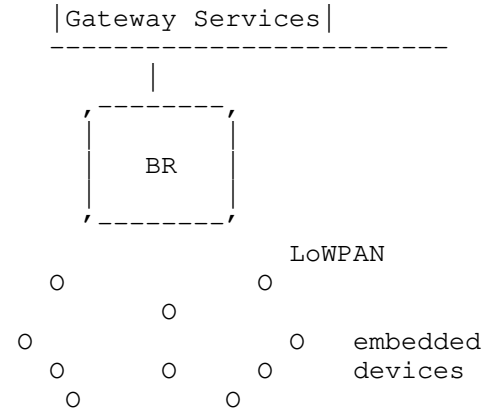


Figure 1: IoT Stub Network

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

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.

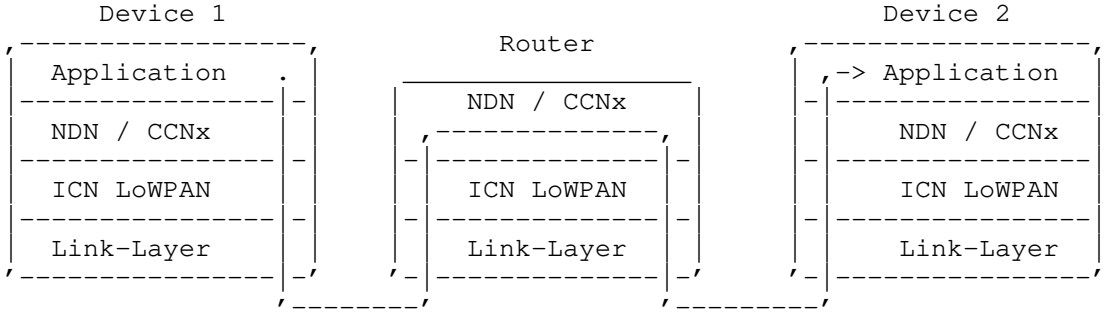


Figure 2: ICN LoWPAN convergence layer for IEEE 802.15.4

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 variable-sized 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 mandatory and 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 codomain 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) and is missing the type, length and the value field.

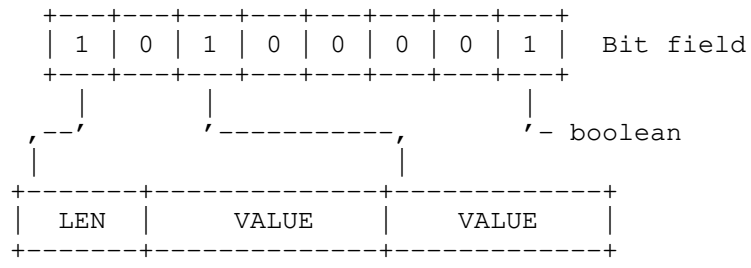


Figure 3: Compression using a compact bit field to encode context information.

3.3. Stateful Header Compression

ICN LoWPAN further employs 2 stateful compression schemes to enhance size reductions. These mechanisms rely on shared contexts that are either distributed and maintained in the whole LoWPAN, or are generated on-demand for a particular Interest-data path.

3.3.1. LoWPAN-local State

A context identifier (CID) is a 1-octet wide number that refers to a particular conceptual context between network devices and MAY be used to replace frequently appearing information, like name prefixes, suffixes, or meta information, such as Interest lifetime.

The initial distribution and maintenance of shared context is out of scope. Frames containing unknown or invalid CIDs are silently discarded.

3.3.2. 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 to the original Name TLV, or they contain the original Name TLV as a prefix. ICN LoWPAN reduces this duplication in responses by replacing Name TLVs with 1-octet wide HopIDs. While an Interest is forwarded, each hop generates an ephemeral HopID that is tied to a PIT entry. Each HopID MUST be unique within the local PIT and only exist 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 4).

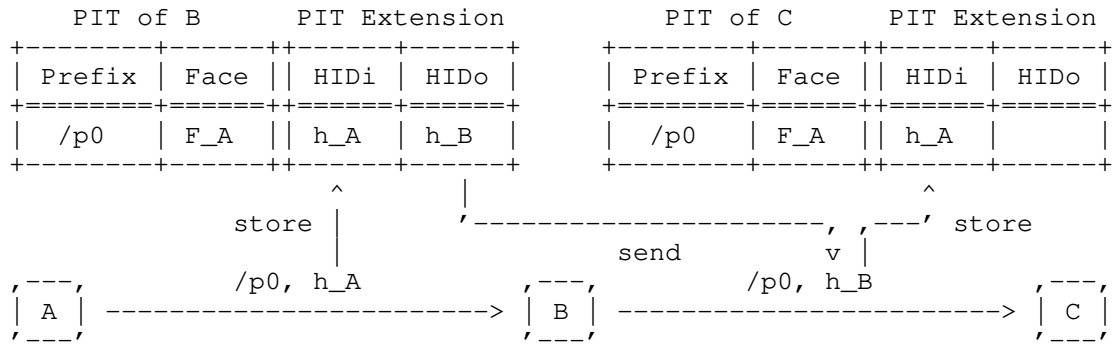


Figure 4: Setting compression state en-route (Interest).

Responses include HopIDs that were obtained from Interests. If the returning Name TLV equals the original Name TLV, then the name is elided fully. Otherwise, the distinct 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 before forwarding the reponse (Figure 5).

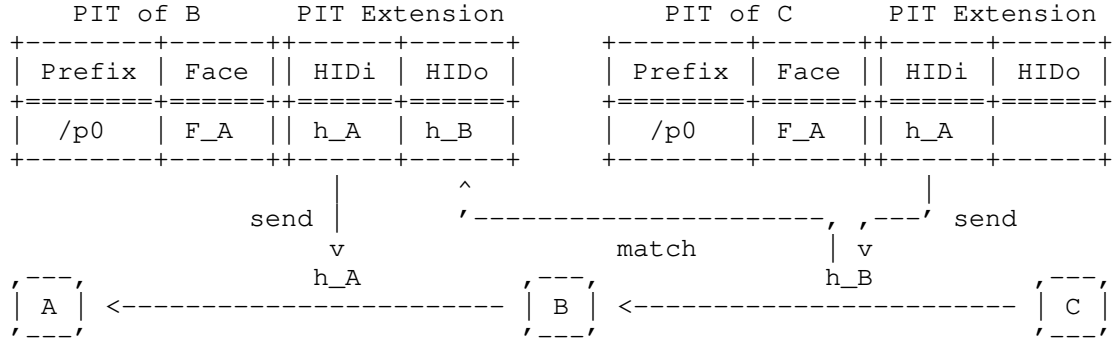


Figure 5: Eliding Name TLVs using en-route state (data).

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 networks coexist on the same link layer. 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 prepend 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 2 ("1111 0010 (0xF2)") for NDN and page 3 ("1111 0011 (0xF3)") for CCNx.

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

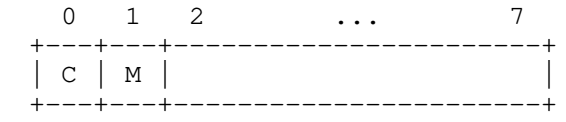


Figure 6: Base dispatch format for NDN

C: Compression

0: The message is uncompressed.

1: The message is compressed.

M: Message Type

0: The payload contains a Interest message.

1: The payload contains a Data message.

The encapsulation format for ICN LoWPAN identifying an NDN Interest message is exemplarily displayed in Figure 7.

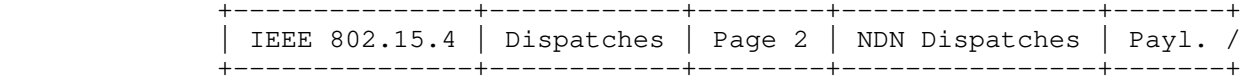


Figure 7: LoWPAN Encapsulation of NDN Interest with ICN LoWPAN

IEEE 802.15.4: The IEEE 802.15.4 header.

Dispatches: Optional additional dispatch types.

Page 2: Page Switch 2 (0xF2) for NDN.

NDN Dispatches: NDN dispatches as defined in Section 5.

Payload: The actual (un-)compressed NDN Interest.

4.2. Link Fragmentation

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 the fragmentation handling of [RFC4944].

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

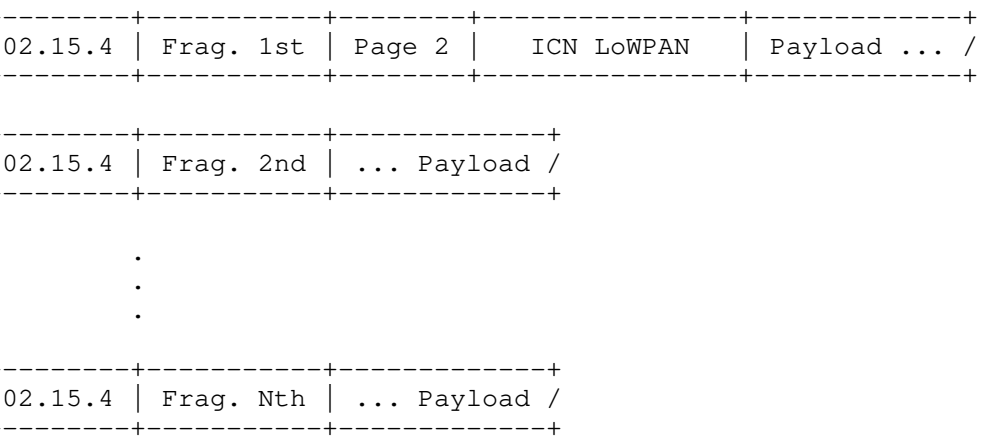


Figure 8: Fragmentation scheme

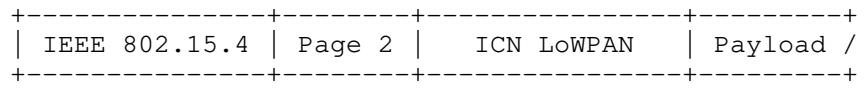


Figure 9: Reassembled ICN LoWPAN frame

4.3. Integrating Stateful Header Compression

4.3.1. LoWPAN-Local State

A CID is appended to the last ICN LoWPAN dispatch octet. Multiple CIDs are chained together, whereas the most significant bit indicates the presence of a subsequent CID (Figure 10).

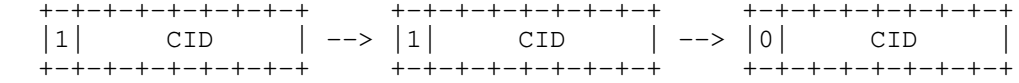


Figure 10: Multiple 1-octet wide context identifiers.

4.3.2. En-Route State

The HopID is included as the very first CID. To distinguish the HopID from a typical LoWPAN-local CID, the 1st bit MUST be set (Figure 11). This yields 64 distinct HopIDs. If this range (0..63) is exhausted, the messages MUST be sent without en-route state compression until new HopIDs are available.

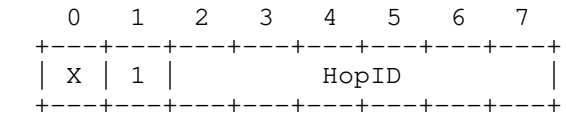


Figure 11: Context Identifier as HopID.

5. ICN LoWPAN 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 octets. Figure 12 shows the NDN TLV encoding scheme.

If the type or length number is less than "253", then that number is encoded into the actual type or length field (Figure 12 a). If the number is greater or equals "253" and fits into 2 octets, then the type or lengh field is set to "253" and the number is encoded in the next following 2 octets in network byte order, i.e., from the most significant byte (MSB) to the least significant byte (LSB) (Figure 12 b). If the number is greater than 2 octets and fits into 4 octets, then the type or length field is set to "254" and the number is encoded in the subsequent 4 octets in network byte order (Figure 12 c). For greater numbers, the type or length field is set to "255" and the number is encoded in the subsequent 8 octets in network byte order (Figure 12 d).

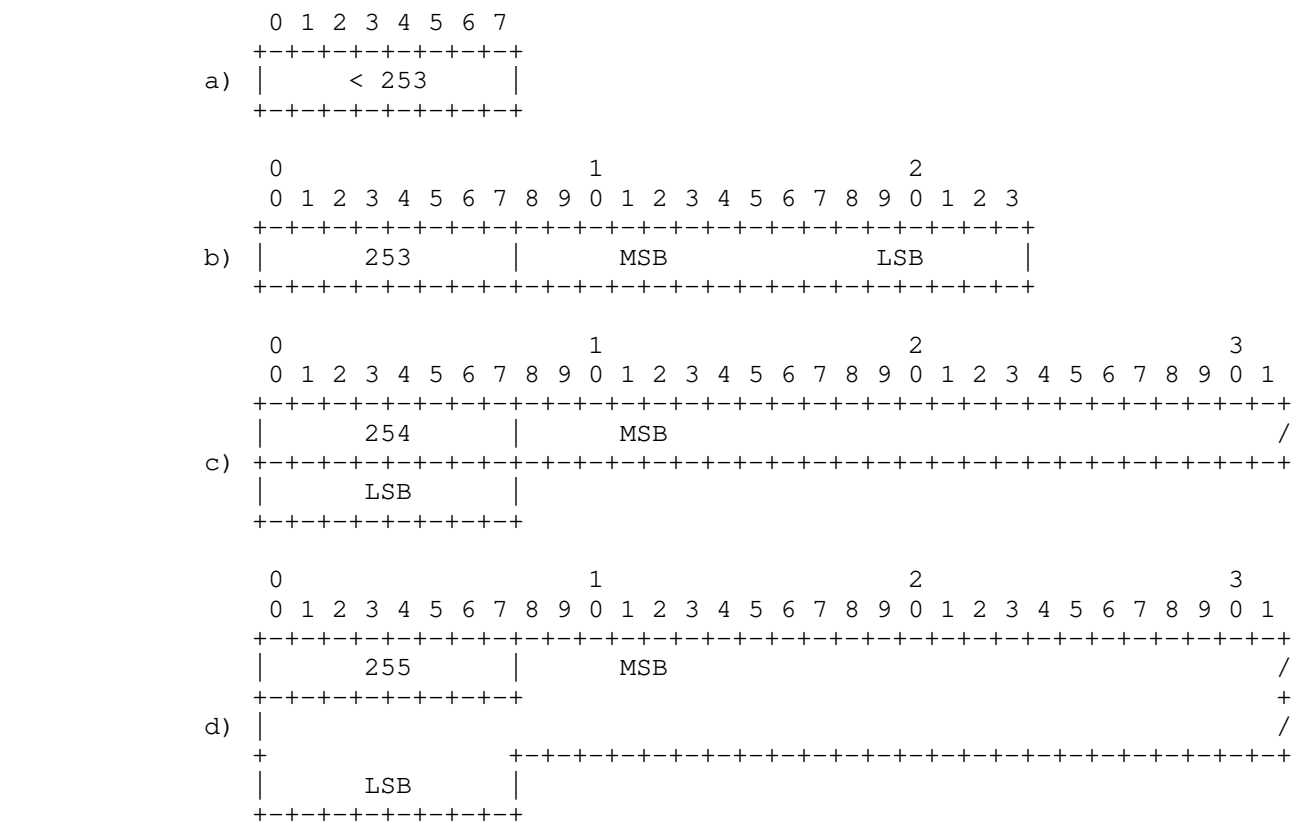


Figure 12: NDN TLV encoding scheme

In this document, compressed NDN TLVs make use of a different TLV scheme that puts more emphasis on size reduction. Instead of using the first octet as a marker for the number of following octets, the compressed NDN TLV scheme uses a method to chain a variable number of octets together. If an octet equals "255 (0xFF)", then the following octet will also be interpreted. The actual value of a chain equals the sum of all links.

If the type or length number is less than "255", then that number is encoded into the actual type or length field (Figure 13 a). If the type or length number (X) fits into 2 octets, then the first octet is set to "255" and the subsequent octet equals "X mod 255" (Figure 13 b). Following this scheme, a variable-sized number (X) is encoded using multiple octets of "255" with a trailing octet containing "X mod 255" (Figure 13 c).

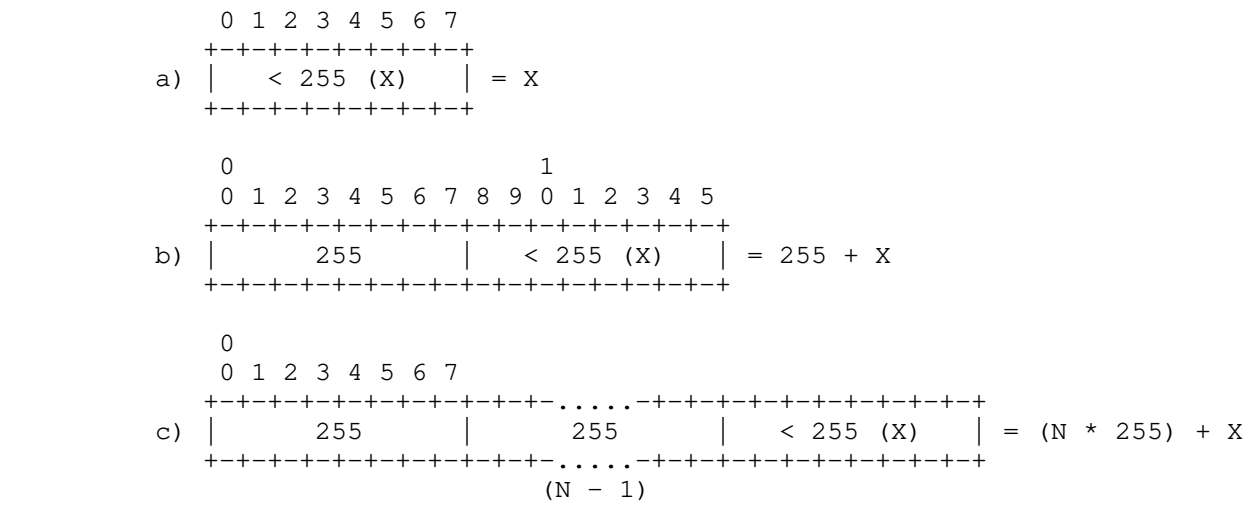


Figure 13: Compressed NDN TLV encoding scheme

5.2. Name TLV Compression

This Name TLV compression encodes length fields of two consecutive NameComponent TLVs into one octet, using 4 bits each. This process limits the length of a NameComponent TLV to 15 octets. A length of 0 marks the end of the compressed Name TLV.

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

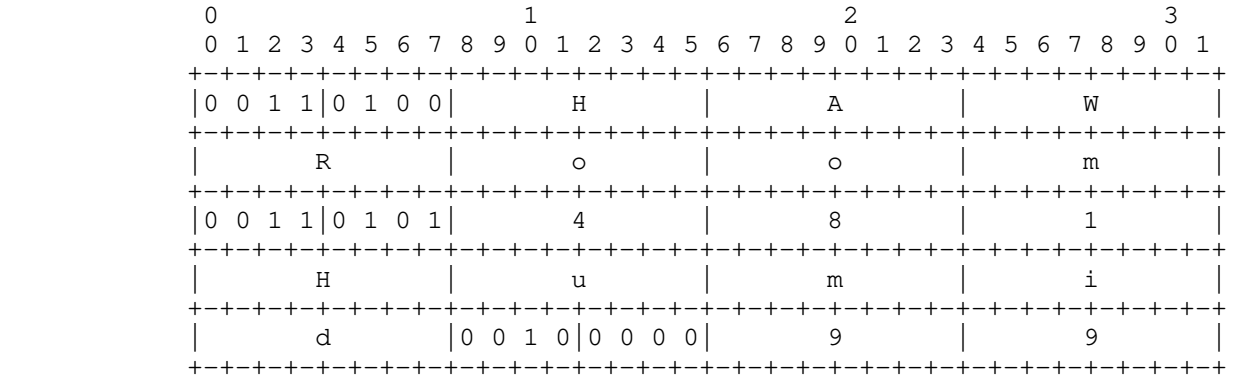


Figure 14: Name TLV compression for /HAW/Room/481/Humid/99

5.3. Interest Messages

5.3.1. Uncompressed Interest Messages

An uncompressed Interest message uses the base dispatch format (see Figure 6) and sets the C as well as the M flag to "0" (Figure 15). "resv" MUST be set to 0. The Interest message is handed to the NDN network stack without modifications.

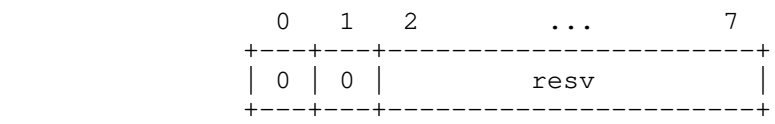


Figure 15: Dispatch format for uncompressed NDN Interest messages

5.3.2. Compressed Interest Messages

The compressed Interest message uses the base dispatch format and sets the C flag to "1" and the M flag to "0". By default, the Interest 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. Otherwise, the message MUST be sent uncompressed.

3. The InterestLifetime TLV length is set to 2. Messages with lifetimes that require more than 2 octets MUST be sent uncompressed.

4. The Nonce TLV, InterestLifetime TLV and HopLimit TLV MUST be moved to the end of the compressed Interest, keeping the order 1) Nonce TLV, 2) InterestLifetime TLV and 3) HopLimit TLV.

5. The Type and Length fields of Nonce TLV, InterestLifetime TLV and HopLimit TLV are elided. The presence of each TLV is deduced from the remaining length to parse. The Nonce TLV has a fixed length of 4, the InterestLifetime TLV has a fixed length of 2 and the HopLimit TLV has a fixed length of 1. Any combination yields a distinct value that matches the remaining length to parse.

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 16.

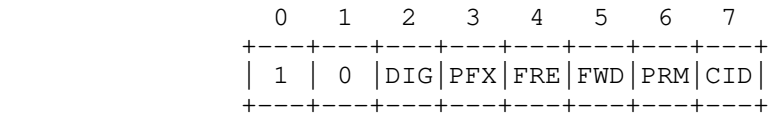


Figure 16: Dispatch format for compressed NDN Interest messages

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.

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.

PRM: Parameters TLV

0: The uncompressed message does not include a Parameters TLV.

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

CID: Context Identifiers

0: CID(s) are not appended to the dispatch octet.

1: CID(s) are appended to the dispatch octet.

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" and the M flag to "1" (Figure 17). "resv" MUST be set to 0. The Data message is handed to the NDN network stack without modifications.

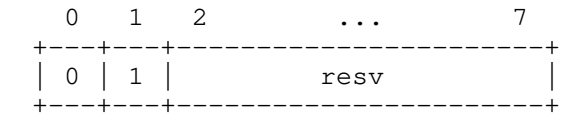


Figure 17: Dispatch format for uncompressed NDN Data messages

5.4.2. Compressed Data Messages

The compressed Data message uses the base dispatch format and sets the C flag as well as the M flag to "1". 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. Otherwise, the message MUST be sent uncompressed.

3. The MetaInfo Type and Length fields are elided from the compressed Data message.

4. If present, the FinalBlockId TLV is encoded according to Section 5.2.

5. The ContentType TLV length is set to 1. Messages with ContentTypes that require more than 1 octet MUST be sent uncompressed.

6. The FreshnessPeriod TLV length is set to 2. Messages with FreshnessPeriods that require more than 2 octets MUST be sent uncompressed.

7. The FreshnessPeriod TLV and ContntType TLV MUST be moved to the end of the compressed Data, keeping the order 1) FreshnessPeriod TLV and 2) ContentType TLV.

8. The Type and Length fields of ContentType TLV and FreshnessPeriod TLV are elided. The presence of each TLV is deduced from the remaining length to parse. The FreshnessPeriod TLV has a fixed length of 2 and the ContentType TLV has a fixed length of 1. Any combination yields a distinct value that matches the remaining length to parse.

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 18.

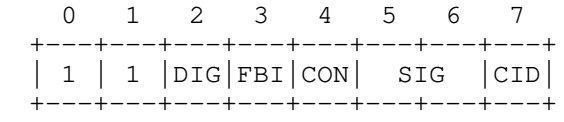


Figure 18: Dispatch format for compressed NDN Data messages

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.

FBI: FinalBlockId TLV

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

1: The uncompressed message does include a FinalBlockId.

CON: Content TLV

0: The uncompressed message does not include a Content TLV.

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

SIG: Signature TLV

00: The Type fields of the SignatureInfo TLV, SignatureType TLV and SignatureValue TLV are removed.

01: Reserved.

10: Reserved.

11: Reserved.

CID: Context Identifiers

0: CID(s) are not appended to the dispatch octet.

1: CID(s) are appended to the dispatch octet.

6. ICN LoWPAN for CCNx

6.1. TLV Encoding

The CCNx TLV encoding is described in [I-D.irtf-icnrg-ccnxmessages]. Type and Length fields are of fixed length of 2 octets each.

In this document, the TLV encoding is changed to the more space efficient encoding described in Section 5.1. Type and Length fields MUST be encoded as in Figure 13.

6.2. Name TLV Compression

Name TLVs are compressed using the same approach outlined in Section 5.2.

6.3. Interest Messages

6.3.1. Uncompressed Interest Messages

An uncompressed Interest message uses the base dispatch format (see Figure 6) and sets the C as well as the M flag to "0" (Figure 19). "resv" MUST be set to 0. The Interest message is handed to the CCNx network stack without modifications.

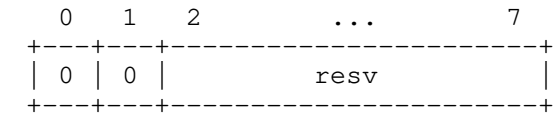


Figure 19: Dispatch format for uncompressed CCNx Interest messages

6.3.2. Compressed Interest Messages

The compressed Interest message uses the base dispatch format and sets the C flag to "1" and the M flag to "0". By default, the Interest message is compressed with the following base rule set:

1. The Type and Length fields of the CCNx Message TLV are elided and are obtained from the Fixed Header on decompression.

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 20.

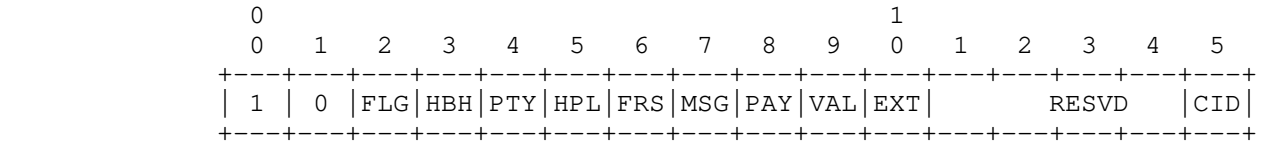


Figure 20: Dispatch format for compressed CCNx Interest messages

FLG: Flags field in the Fixed Header

- 0: The Flags field equals 0 and is removed from the Interest message.
- 1: The Flags field is carried in-line.

HBH: Optional Hop-By-Hop Header TLVs

- 0: No Hop-By-Hop Header TLVs are present in the Interest message. Also, the HeaderLength field in the fixed header is elided from the Interest message and assumed to be "8".
- 1: Hop-By-Hop Header TLVs are present in the Interest message. An additional octet follows immediately that handles Hop-By-Hop Header TLV compressions and is described in Section 6.3.3.

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 is carried in-line

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

FRS: Reserved field in the fixed header

0: The Reserved field is carried in-line

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

MSG: Optional Interest Message TLVs

0: No Interest Message TLVs are present in the Interest message.

1: Interest Message TLVs are present in the Interest message. An additional octet follows immediately that handles Interest Message TLV compressions and is described in Section 6.3.4.

PAY: Optional Payload TLV

0: The Payload TLV is absent.

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

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 octet follows immediately that handles validation related TLV compressions and is described in Section 6.3.5.

EXT: Extension

0: No extension octet follows.

1: An extension octet follows immediately. Extension octets are used to extend the compression scheme, but are out of scope of this document.

CID: Context Identifiers

0: CID(s) are not appended to the last dispatch octet.

1: CID(s) are appended to the last dispatch octet.

6.3.3. 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 [I-D.irtf-icnrg-ccnxmessages], but several more can be defined in higher level specifications. For better compression, an ordering of Hop-By-Hop TLVs is enforced as follows:

1. Interest Lifetime TLV

2. Message Hash TLV

With this ordering in place, Type fields are elided from the Interest Lifetime TLV and the Message Hash TLV.

Note: If the original Interest message includes Hop-By-Hop Header TLVs with a different ordering, then they remain uncompressed.

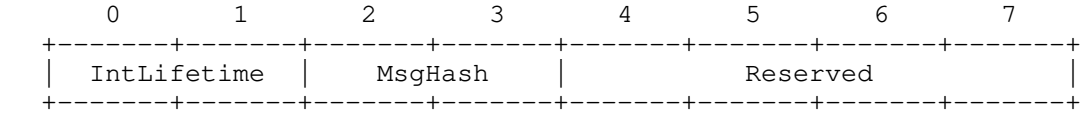


Figure 21: Dispatch for HBH Compression

IntLifetime: InterstLifetime Hop-By-Hop Header TLV

00: The Interest Lifetime TLV is absent.

01: The Interest Lifetime TLV is present and the type field is removed.

10: The Interest Lifetime TLV is absent and a default value of 0 seconds is assumed.

11: The Interest Lifetime TLV is absent and a default value of 10 minutes is assumed.

MsgHash: Message Hash Hop-By-Hop Header TLV

00: The Message Hash TLV is absent.

01: The Message Hash 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 octets. The outer Message Hash 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 octets. The outer Message Hash TLV is omitted.

6.3.4. Interest Message TLVs Compression

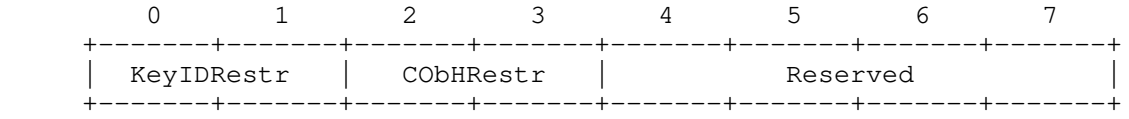


Figure 22: Dispatch for Interest Messages

KeyIDRestr: Optional KeyIdRestriction TLV within a CCNx Message TLV

00: The KeyIdRestriction TLV is absent.

01: The KeyIdRestriction 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 octets. The outer KeyIdRestriction 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 octets. The outer KeyIdRestriction TLV is omitted.

CObHRestr: Optional ContentObjectHashRestriction TLV within a CCNx Message TLV

00: The ContentObjectHashRestriction TLV is absent.

01: The ContentObjectHashRestriction 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 octets. The outer ContentObjectHashRestriction 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 octets. The outer ContentObjectHashRestriction TLV is omitted.

6.3.5. Validation

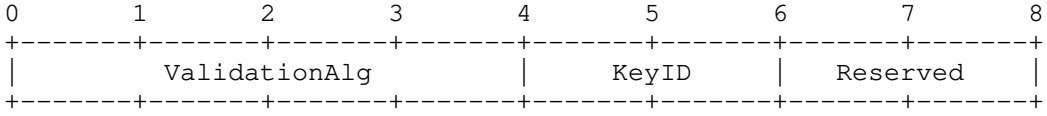


Figure 23: Dispatch for Interset Validations

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 inclued. 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 octets. 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 octets. The outer KeyId TLV is omitted.

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

6.4. Content Objects

6.4.1. Uncompressed Content Objects

An uncompressed Content object uses the base dispatch format (see Figure 6) and sets the C flag to "0" and the M flag to "1" (Figure 24). "resv" MUST be set to 0. The Content object is handed to the CCNx network stack without modifications.

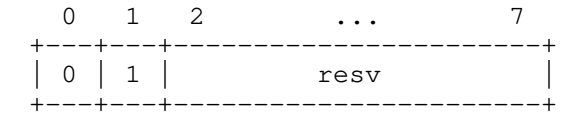


Figure 24: Dispatch format for uncompressed CCNx Content objects

6.4.2. Compressed Content Objects

The compressed Content object uses the base dispatch format and sets the C flag as well as the M flag to "1". By default, the Content object is compressed with the following base rule set:

1. The PacketType field is elided from the Fixed Header.

2. The Type and Length fields of the CCNx Message TLV are elided and are obtained from the Fixed Header on decompression.

Further TLV compression is indicated by the ICN LoWPAN dispatch in Figure 25.

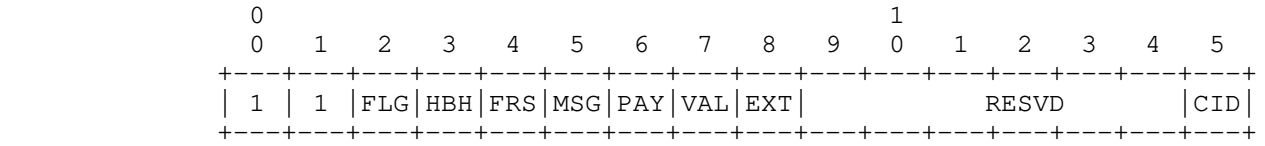


Figure 25: Dispatch format for compressed CCNx Content objects

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

HBH: Optional Hop-By-Hop Header TLVs

- 0: No Hop-By-Hop Header TLVs are present in the Content Object message. Also, the HeaderLength field in the fixed header is elided from the Content Object message and assumed to be "8".

- 1: Hop-By-Hop Header TLVs are present in the Content Object message. An additional octet follows immediately that handles Hop-By-Hop Header TLV compressions and is described in Section 6.4.3.

FRS: Reserved field in the Fixed Header See Section 6.3.2.

MSG: Optional Content Object Message TLVs

- 0: No Content Object Message TLVs are present in the Content Object message.

- 1: Content Object Message TLVs are present in the Content Object message. An additional octet follows immediately that handles Content Object Message TLV compressions and is described in Section 6.4.4.

PAY: Optional Payload TLV See Section 6.3.2.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs See Section 6.3.2.

EXT: Extension See Section 6.3.2.

CID: Context Identifiers

0: CID(s) are not appended to the last dispatch octet.

1: CID(s) are appended to the last dispatch octet.

6.4.3. 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 [I-D.irtf-icnrg-ccnxmessages], but several more can be defined in higher level specifications. For better compression, an ordering of Hop-By-Hop TLVs is enforced as follows:

1. Recommended Cache Time TLV

2. Message Hash TLV

With this ordering in place, Type fields are elided from the Recommended Cache Time TLV and Message Hash TLV.

Note: If the original Content Object message includes Hop-By-Hop Header TLVs with a different ordering, then they remain uncompressed.

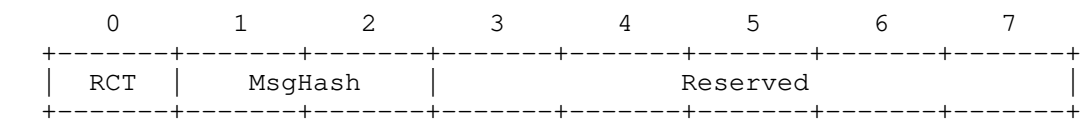


Figure 26: Dispatch for HBH Compression

RCT: Recommended Cache Time Hop-By-Hop Header 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.

MsgHash: Message Hash Hop-By-Hop Header TLV See Section 6.3.3.

6.4.4. Content Object Message TLVs Compression

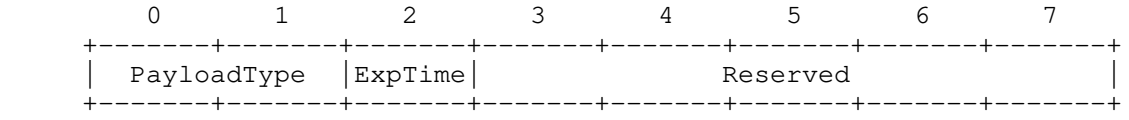


Figure 27: Dispatch for Message TLVs

PayloadType: Optional PayloadType TLV within a CCNx Message TLV

- 00: The PayloadType TLV is absent and T_PAYLOADTYPE_DATA is assumed.
- 01: The PayloadType TLV is absent and T_PAYLOADTYPE_KEY is assumed.
- 10: The PayloadType TLV is absent and T_PAYLOADTYPE_LINK is assumed.
- 11: The PayloadType TLV is present and uncompressed.

ExpTime: Optional ExpiryTime TLV within a CCNx Message TLV

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

7. Security Considerations

TODO

8. IANA Considerations

8.1. Page Switch Dispatch Type

This document makes use of "Page 2" from the existing paging dispatches in [RFC8025].

9. References

9.1. Normative References

[ieee802.15.4]
IEEE Computer Society, "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>.

[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>.

9.2. Informative References

[CCN-LITE] "CCN-lite: A lightweight CCNx and NDN implementation", <http://ccn-lite.net/>.

[I-D.irtf-icnrg-ccnxmessages] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-08 (work in progress), July 2018.

[I-D.irtf-icnrg-ccnxsemantics] Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", draft-irtf-icnrg-ccnxsemantics-09 (work in progress), June 2018.

[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-EXP] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, TC., and M. Waehlisch, "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-MAC] Kietzmann, P., Gundogan, C., Schmidt, TC., Hahm, O., and M. Waehlisch, "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", <http://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>.

[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. Waehlisch, "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>.

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 octet 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 28 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 octets.

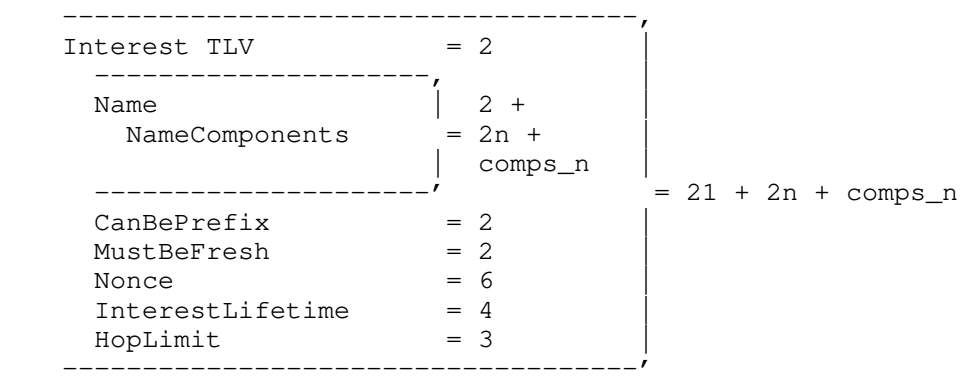


Figure 28: Estimated size of an uncompressed NDN Interest

Figure 29 depicts the size requirements after compression.

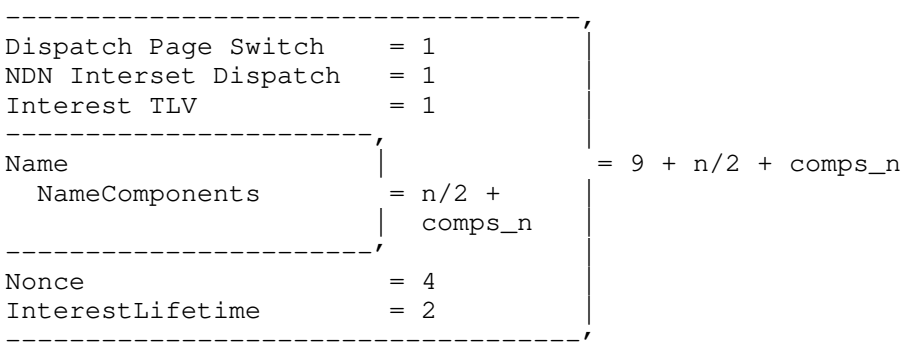


Figure 29: Estimated size of a compressed NDN Interest

The size difference is:
12 + 1.5n octets.

For the name "/DE/HH/HAW/BT7", the total size gain is 18 octets, which is 46% of the uncompressed packet.

A.1.2. Data

Figure 30 depicts the size requirements for a basic, uncompressed NDN Data containing a FreshnessPeriod as MetaInfo. A FreshnessPeriod of 1 minute is assumed. The value is thereby encoded using 2 octets. An HMACWithSha256 is assumed as signature. The key locator is assumed to contain a Name TLV of length klen.

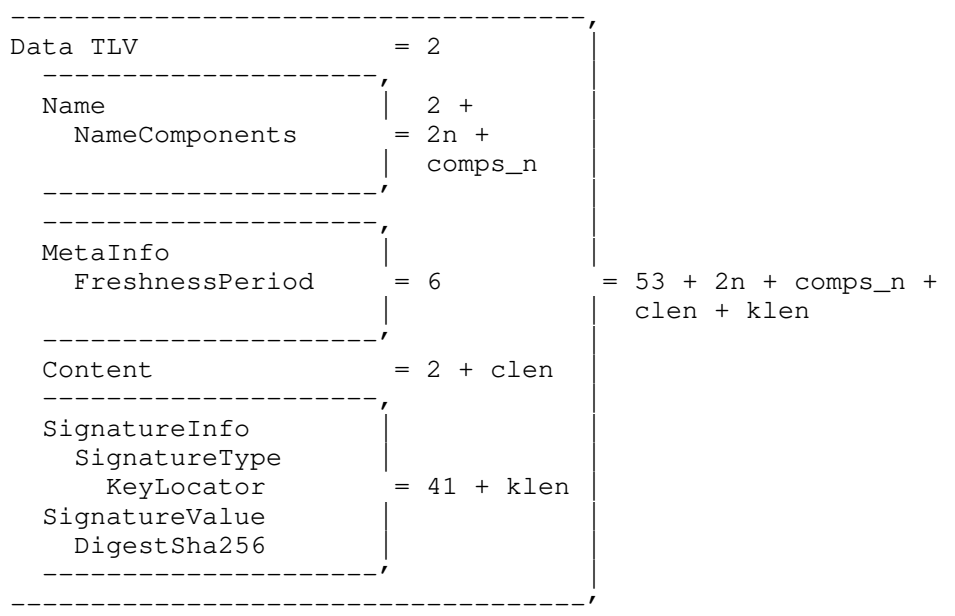


Figure 30: Estimated size of an uncompressed NDN Data

Figure 31 depicts the size requirements for the compressed version of the above Data packet.

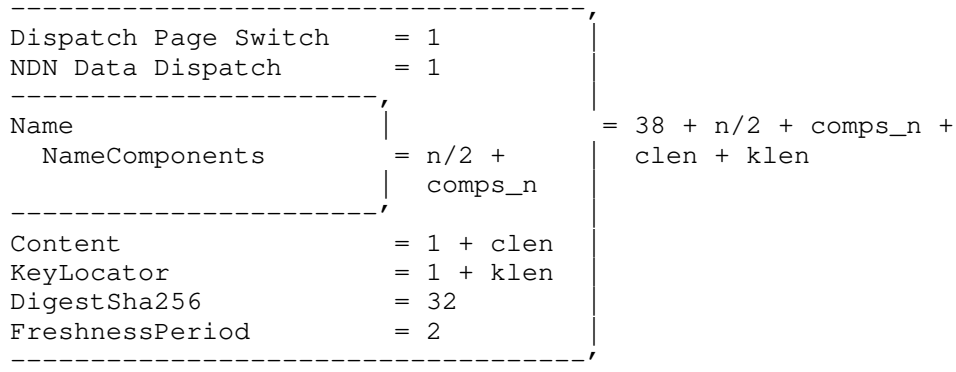


Figure 31: Estimated size of a compressed NDN Data

The size difference is:
15 + 1.5n octets.

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

A.2. CCNx

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

A.2.1. Interest

Figure 32 depicts the size requirements for a basic, uncompressed CCNx Interest. No Hop-By-Hop TLVs are included and the protocol version as well as the reserved field are assumed to be 0. A KeyIdRestriction TLV with T_SHA-256 is included to limit the responses to Content Objects containing the specific key.

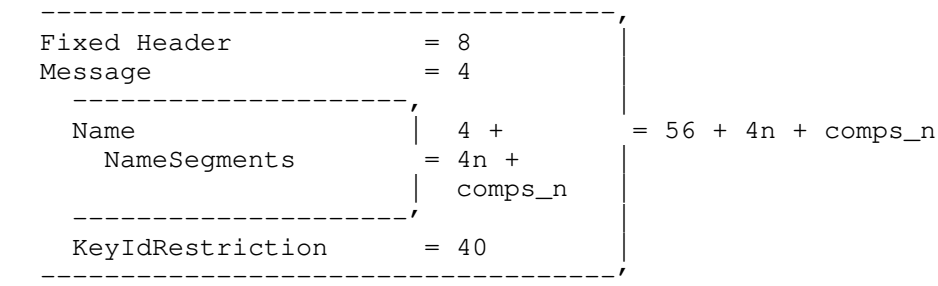


Figure 32: Estimated size of an uncompressed CCNx Interest

Figure 33 depicts the size requirements after compression.

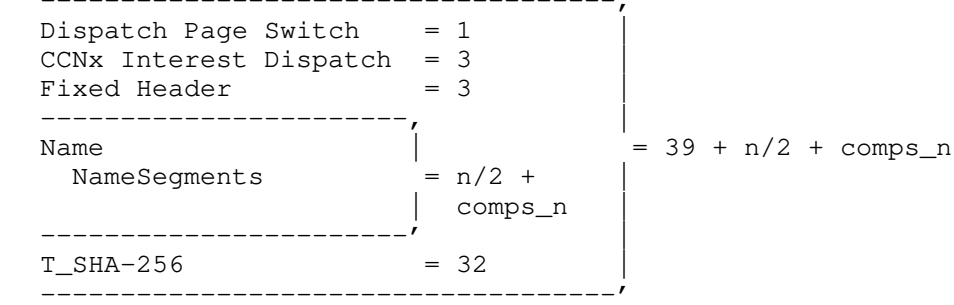


Figure 33: Estimated size of a compressed CCNx Interest

The size difference is:
17 + 3.5n octets.

For the name "/DE/HH/HAW/BT7", the total size gain is 31 octets, which is 38% of the uncompressed packet.

A.2.2. Data

Figure 34 depicts the size requirements for a basic, uncompressed CCNx Data containing an ExpiryTime Message TLV, an HMAC_SHA-256 signature, the signature time and a hash of the shared secret key.

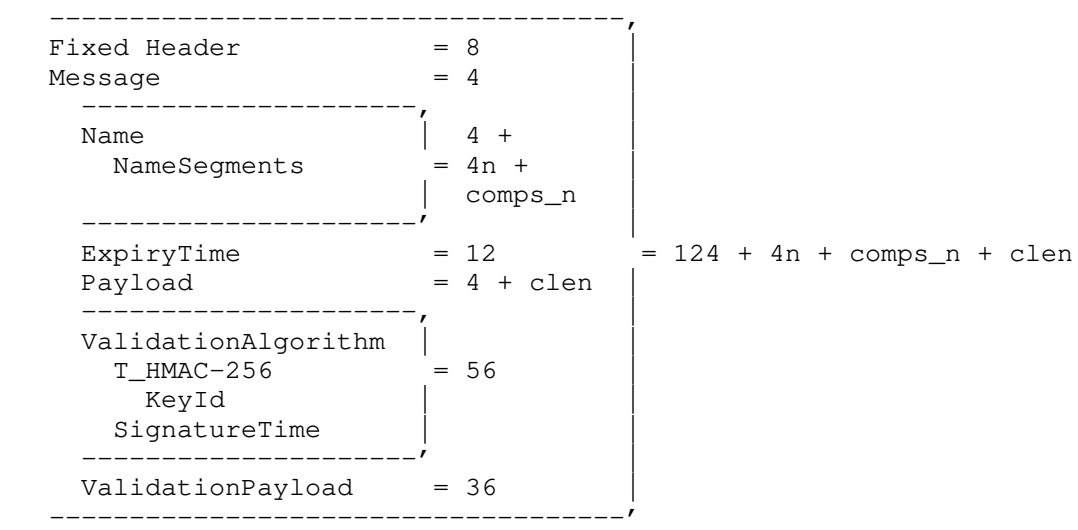


Figure 34: Estimated size of an uncompressed CCNx Data Object

Figure 35 depicts the size requirements for a basic, compressed CCNx Data.

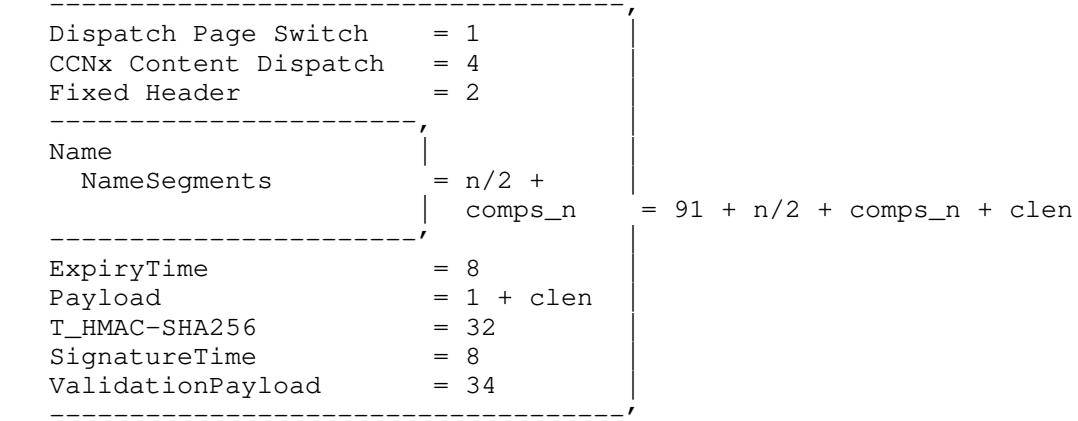


Figure 35: Estimated size of a compressed CCNx Data Object

The size difference is:
33 + 3.5n octets.

For the name "/DE/HH/HAW/BT7", the total size gain is 47 octets.

Acknowledgments

Authors' Addresses

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

EMail: t.schmidt@haw-hamburg.de
URI: http://inet.haw-hamburg.de/members/schmidt

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

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

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

EMail: christopher.scherb@unibas.ch

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

EMail: claudio.marxer@unibas.ch

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

EMail: christian.tschudin@unibas.ch

ICNRG
Internet-Draft
Intended status: Experimental
Expires: July 28, 2019

M. Mosko
PARC, Inc.
I. Solis
LinkedIn
C. Wood
University of California Irvine

January 24, 2019

CCNx Messages in TLV Format
draft-irtf-icnrg-ccnxmessages-09

Abstract

This document specifies the encoding of CCNx messages in a TLV packet format, including the TLV types used by each message element and the encoding of each value. The semantics of CCNx messages follow the encoding-independent CCNx Semantics specification.

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 http://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 July 28, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a Type-Length-Value (TLV) packet format and the TLV type and value encodings for CCNx messages. A full description of the CCNx network protocol, providing an encoding-free description of CCNx messages and message elements, may be found in [CCNSemantics]. CCNx is a network protocol that uses a hierarchical name to forward requests and to match responses to requests. It does not use endpoint addresses, such as Internet Protocol. Restrictions in a request can limit the response by the public key of the response's signer or the cryptographic hash of the response. Every CCNx forwarder along the path does the name matching and restriction checking. The CCNx protocol fits within the broader framework of Information Centric Networking (ICN) protocols [RFC7927].

This document describes a TLV scheme using a fixed 2-byte T and a fixed 2-byte L field. The rational for this choice is described in Section 5. Briefly, this choice avoids multiple encodings of the same value (aliases) and reduces the work of a validator to ensure compliance. Unlike some uses of TLV in networking, the each network hop must evaluate the encoding, so even small validation latencies at each hop could add up to a large overall forwarding delay. For very small packets or low throughput links, where the extra bytes may become a concern, one may use a TLV compression protocol, for example [compress] and [CCNxz].

This document specifies:

- The TLV packet format.
- The overall packet format for CCNx messages.
- The TLV types used by CCNx messages.
- The encoding of values for each type.
- Top level types that exist at the outermost containment.

- Interest TLVs that exist within Interest containment.

- Content Object TLVs that exist within Content Object containment.

This document is supplemented by this document:

- Message semantics: see [CCNSemantics] for the protocol operation regarding Interest and Content Object, including the Interest Return protocol.

- URI notation: see [CCNxURI] for the CCNx URI notation.

The type values in Section 4 represent the values in common usage today. These values may change pending IANA assignments. All type values are relative to their parent containers. For example, each level of a nested TLV structure might define a "type = 1" with a completely different meaning. In the following, we use the symbolic names defined in that section.

Packets are represented as 32-bit wide words using ASCII art. Due to the nested levels of TLV encoding and the presence of optional fields and variable sizes, there is no concise way to represent all possibilities. We use the convention that ASCII art fields enclosed by vertical bars "|" represent exact bit widths. Fields with a forward slash "/" are variable bit widths, which we typically pad out to word alignment for picture readability.

The document represents the consensus of the ICN RG. It is the first ICN protocol from the RG, created from the early CCNx protocol [nnc] with significant revision and input from the ICN community and RG members. The draft has received critical reading by several members of the ICN community and the RG. The authors and RG chairs approve of the contents. The document is sponsored under the IRTF and is not issued by the IETF and is not an IETF standard. This is an experimental protocol and may not be suitable for any specific application and the specification may change in the future.

1.1. 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].

2. Definitions

- Name: A hierarchically structured variable length identifier. It is an ordered list of path segments, which are variable length octet strings. In human-readable form, it is represented in URI

format as ccnx:/path/part. There is no host or query string. See [CCNxURI] for complete details.

- Interest: A message requesting a Content Object with a matching Name and other optional selectors to choose from multiple objects with the same Name. Any Content Object with a Name and attributes that matches the Name and optional selectors of the Interest is said to satisfy the Interest.

- Content Object: A data object sent in response to an Interest request. It has an optional Name and a content payload that are bound together via cryptographic means.

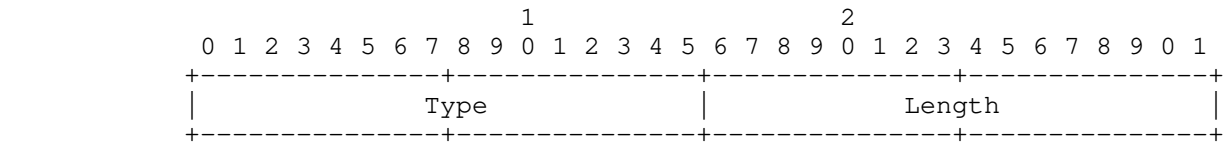
3. Type-Length-Value (TLV) Packets

We use 16-bit Type and 16-bit Length fields to encode TLV based packets. This provides 64K different possible types and value field lengths of up to 64KiB. With 64K possible types at each level of TLV encoding, there should be sufficient space for basic protocol types, while also allowing ample room for experimentation, application use, vendor extensions, and growth. This encoding does not allow for jumbo packets beyond 64 KiB total length. If used on a media that allows for jumbo frames, we suggest defining a media adapation envelope that allows for multiple smaller frames.

There are several global TLV definitions that we reserve at all hierarchical contexts. The TLV types in the range 0x1000 - 0x1FFF are reserved for experimental use. The TLV type T_ORG is also reserved for vendor extensions (see Section 3.3.2). The TLV type T_PAD is used to optionally pad a field out to some desired alignment.

Abbrev	Name	Description
T_ORG	Vendor Specific Information (Section 3.3.2)	Information specific to a vendor implementation (see below).
T_PAD	Padding (Section 3.3.1)	Adds padding to a field (see below).
n/a	Experimental	Experimental use.

Table 1: Reserved TLV Types



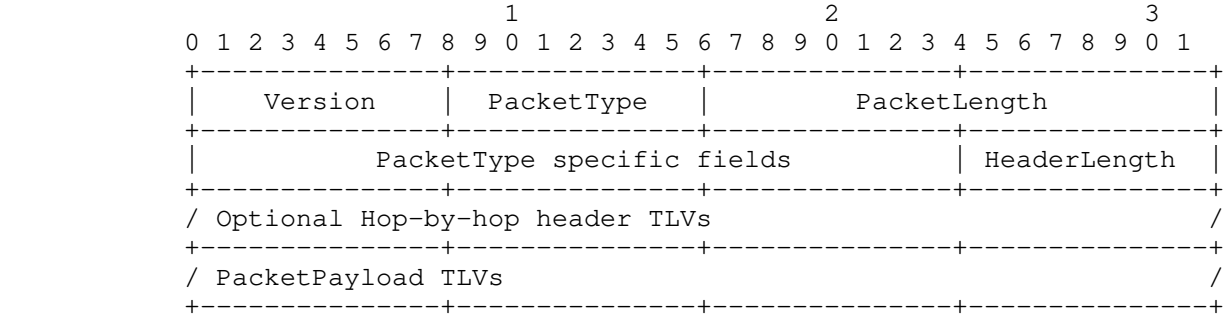
The Length field contains the length of the Value field in octets. It does not include the length of the Type and Length fields. The length MAY be zero.

TLV structures are nestable, allowing the Value field of one TLV structure to contain additional TLV structures. The enclosing TLV structure is called the container of the enclosed TLV.

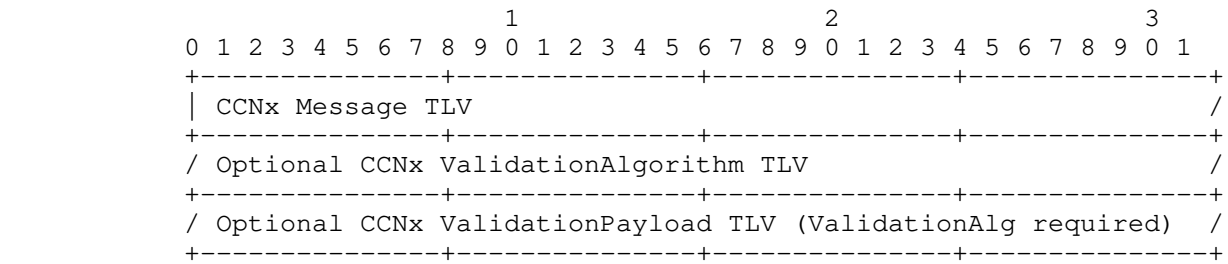
Type values are context-dependent. Within a TLV container, one may re-use previous type values for new context-dependent purposes.

3.1. Overall packet format

Each packet includes the 8 byte fixed header, described below, followed by a set of TLV fields. These fields are optional hop-by-hop headers and the Packet Payload.



The packet payload is a TLV encoding of the CCNx message, followed by optional Validation TLVs.



This document describes the Version "1" TLV encoding.

After discarding the fixed and hop-by-hop headers the remaining PacketPayload should be a valid protocol message. Therefore, the PacketPayload always begins with 4 bytes of type-length that specifies the protocol message (whether it is an Interest, Content Object, or other message type) and its total length. The embedding of a self-sufficient protocol data unit inside the fixed and hop-by-hop headers allows a network stack to discard the headers and operate only on the embedded message. It also de-couples the PacketType field -- which specifies how to forward the packet -- from the PacketPayload.

The range of bytes protected by the Validation includes the CCNx Message and the ValidationAlgorithm.

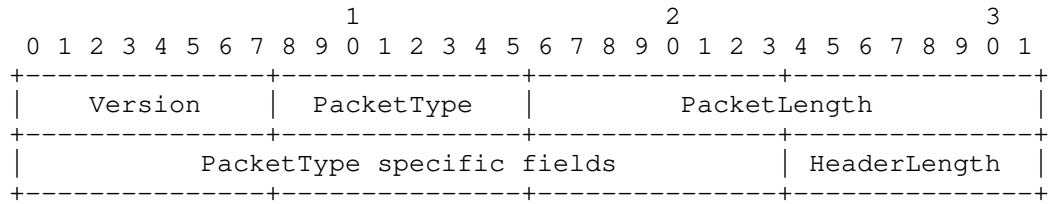
The ContentObjectHash begins with the CCNx Message and ends at the tail of the packet.

3.2. Fixed Headers

CCNx messages begin with an 8 byte fixed header (non-TLV format). The HeaderLength field represents the combined length of the Fixed and Hop-by-hop headers. The PacketLength field represents the entire Packet length from the first byte of Version to the last byte of the packet.

A specific PacketType may assign meaning to the "PacketType specific fields," which are otherwise reserved. For the three defined PacketTypes (Interest, ContentObject, and InterestReturn), we define those values in this document.

The PacketPayload of a CCNx packet is the protocol message itself. The Content Object Hash is computed over the PacketPayload only, excluding the fixed and hop-by-hop headers as those might change from hop to hop. Signed information or Similarity Hashes should not include any of the fixed or hop-by-hop headers. The PacketPayload should be self-sufficient in the event that the fixed and hop-by-hop headers are removed.



- Version: defines the version of the packet.
- HeaderLength: The length of the fixed header (8 bytes) and hop-by-hop headers. The minimum value MUST be "8".
- PacketType: describes forwarder actions to take on the packet.
- PacketLength: Total octets of packet including all headers (fixed header plus hop-by-hop headers) and protocol message.
- PacketType Specific Fields: specific PacketTypes define the use of these bits.

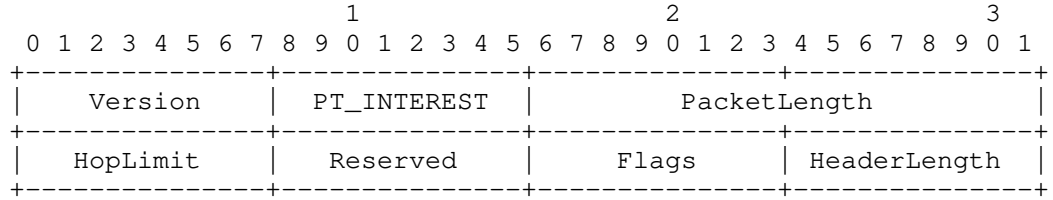
The PacketType field indicates how the forwarder should process the packet. A Request Packet (Interest) has PacketType PT_INTEREST, a Response (Content Object) has PacketType PT_CONTENT, and an InterestReturn has PacketType PT_RETURN.

HeaderLength is the number of octets from the start of the packet (Version) to the end of the hop-by-hop headers. PacketLength is the number of octets from the start of the packet to the end of the packet. Both lengths have a minimum value of 8 (the fixed header itself).

The PacketType specific fields are reserved bits whose use depends on the PacketType. They are used for network-level signaling.

3.2.1. Interest Fixed Header

If the PacketType is PT_INTEREST, it indicates that the PacketPayload should be processed as an Interest message. For this type of packet, the Fixed Header includes a field for a HopLimit as well as Reserved and Flags fields. The Reserved field MUST be set to 0 in an Interest - this field will be set to a return code in the case of an Interest Return. There are currently no Flags defined, so this field MUST be set to 0.



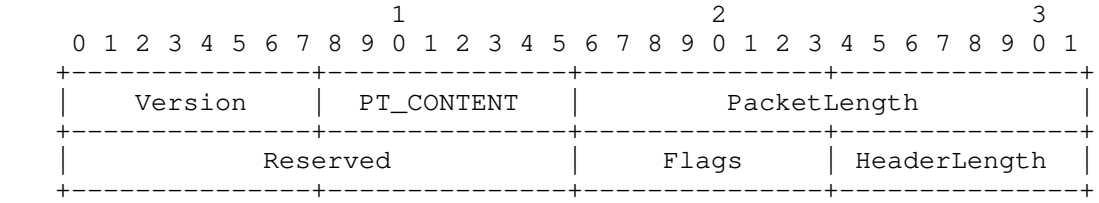
3.2.1.1. Interest HopLimit

For an Interest message, the HopLimit is a counter that is decremented with each hop. It limits the distance an Interest may travel on the network. The node originating the Interest MAY put in any value - up to the maximum of 255. Each node that receives an Interest with a HopLimit decrements the value upon reception. If the value is 0 after the decrement, the Interest MUST NOT be forwarded off the node.

It is an error to receive an Interest with a 0 hop-limit from a remote node.

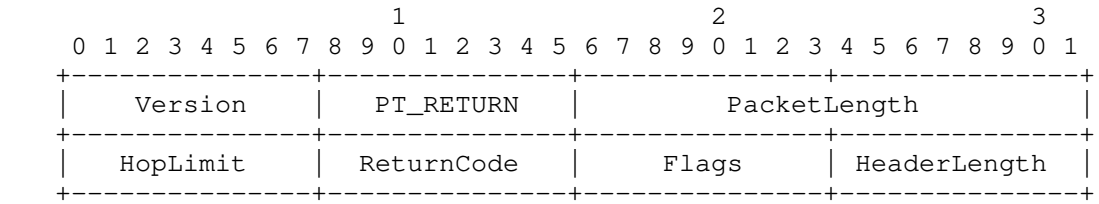
3.2.2. Content Object Fixed Header

If the PacketType is PT_CONTENT, it indicates that the PacketPayload should be processed as a Content Object message. A Content Object defines a Flags field, however there are currently no flags defined, so the Flags field must be set to 0.



3.2.3. InterestReturn Fixed Header

If the PacketType is PT_RETURN, it indicates that the PacketPayload should be processed as a returned Interest message. The only difference between this InterestReturn message and the original Interest is that the PacketType is changed to PT_RETURN and a ReturnCode is is put into the ReturnCode field. All other fields are unchanged from the Interest packet. The purpose of this encoding is to prevent packet length changes so no additional bytes are needed to return an Interest to the previous hop. See [CCNSemantics] for a protocol description of this packet type.



3.2.3.1. InterestReturn HopLimit

This is the original Interest's HopLimit, as received. It is the value before being decremented at the current node (i.e. the received value).

3.2.3.2. InterestReturn Flags

These are the original Flags as set in the Interest.

3.2.3.3. Return Code

The numeric value assigned to the return types is defined below. This value is set by the node creating the Interest Return.

A return code of "0" MUST NOT be used, as it indicates that the returning system did not modify the Return Code field.

Type	Return Type
T_RETURN_NO_ROUTE	No Route
T_RETURN_LIMIT_EXCEEDED	Hop Limit Exceeded
T_RETURN_NO_RESOURCES	No Resources
T_RETURN_PATH_ERROR	Path Error
T_RETURN_PROHIBITED	Prohibited
T_RETURN_CONGESTED	Congested
T_RETURN_MTU_TOO_LARGE	MTU too large
T_RETURN_UNSUPPORTED_HASH_RESTRICTION	Unsupported ContentObjectHashRestriction
T_RETURN_MALFORMED_INTEREST	Malformed Interest

Table 2: Return Codes

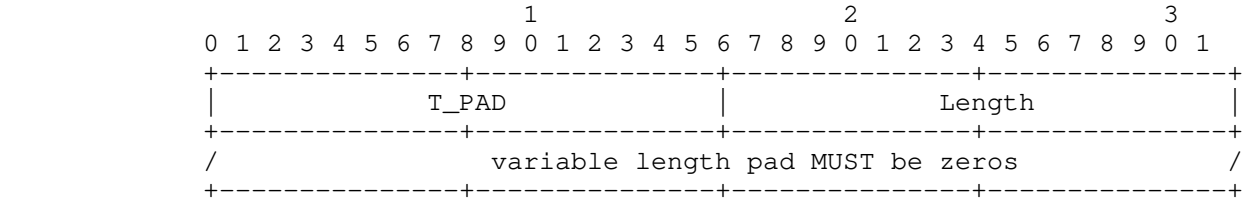
3.3. Global Formats

This section defines global formats that may be nested within other TLVs.

3.3.1. Pad

The pad type may be used by protocols that prefer word-aligned data. The size of the word may be defined by the protocol. Padding 4-byte words, for example, would use a 1-byte, 2-byte, and 3-byte Length. Padding 8-byte words would use a (0, 1, 2, 3, 5, 6, 7)-byte Length.

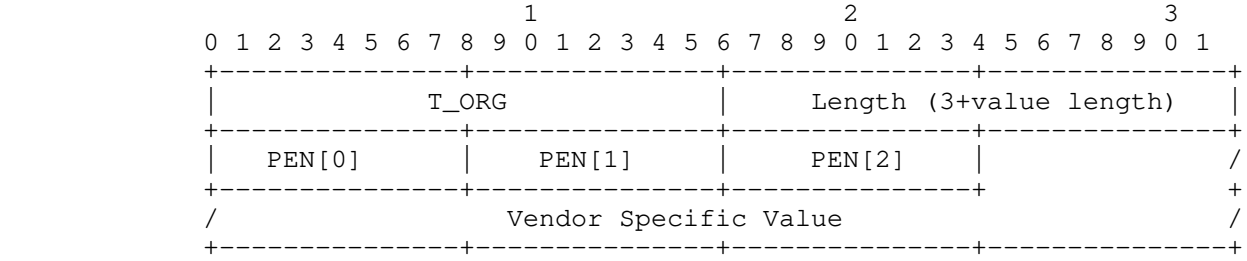
One MUST NOT pad inside a Name. Apart from that, a pad MAY be inserted after any other TLV in the CCNx Message or in the Validation Dependent Data In the remainder of this document, we will not show optional pad TLVs.



3.3.2. Organization Specific TLVs

Organization specific TLVs (also known as Vendor TLVs) MUST use the T_ORG type. The Length field is the length of the organization specific information plus 3. The Value begins with the 3 byte organization number derived from the last three digits of the IANA Private Enterprise Numbers [EpriseNumbers], followed by the organization specific information.

A T_ORG MAY be used as a path segment in a Name, in which case it is a regular path segment and is part of the regular name matching.



3.3.3. Hash Format

Hash values are used in several fields throughout a packet. This TLV encoding is commonly embedded inside those fields to specify the specific hash function used and it's value. Note that the reserved

TLV types are also reserved here for user-defined experimental functions.

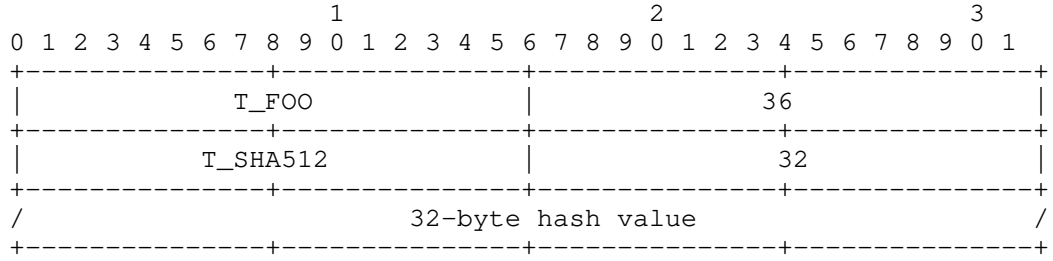
The LENGTH field of the hash value MUST be less than or equal to the hash function length. If the LENGTH is less than the full length, it is taken as the left LENGTH bytes of the hash function output. Only specified truncations are allowed, not arbitrary truncations.

This nested format is used because it allows binary comparison of hash values for certain fields without a router needing to understand a new hash function. For example, the KeyIdRestriction is bit-wise compared between an Interest's KeyIdRestriction field and a ContentObject's KeyId field. This format means the outer field values do not change with differing hash functions so a router can still identify those fields and do a binary comparison of the hash TLV without need to understand the specific hash used. An alternative approach, such as using T_KEYID_SHA512-256, would require each router keep an up-to-date parser and supporting user-defined hash functions here would explode the parsing state-space.

A CCNx entity MUST support the hash type T_SHA-256. An entity MAY support the remaining hash types.

Abbrev	Lengths (octets)
T_SHA-256	32
T_SHA-512	64, 32
n/a	Experimental TLV types

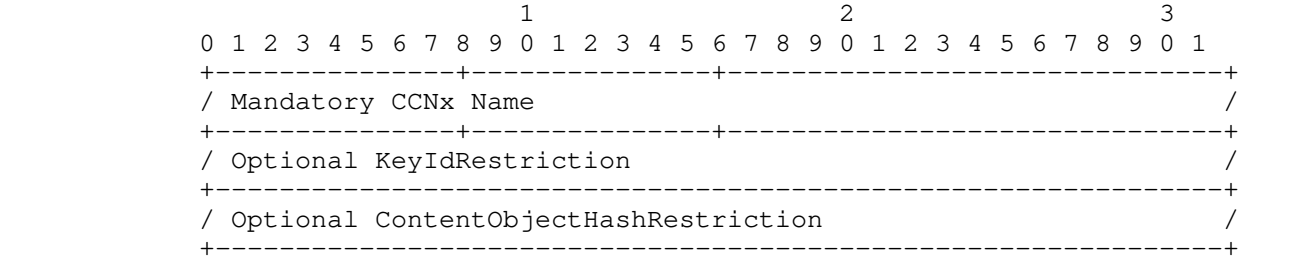
Table 3: CCNx Hash Functions



Example nesting inside type T_FOO

3.3.4. Link

A Link is the tuple: {Name, [KeyIdRestr], [ContentObjectHashRestr]}. It is a general encoding that is used in both the payload of a Content Object with PayloadType = "Link" and in the KeyLink field in a KeyLocator. A Link is essentially the body of an Interest.



3.4. Hop-by-hop TLV headers

Hop-by-hop TLV headers are unordered and meaning MUST NOT be attached to their ordering. Three hop-by-hop headers are described in this document:

Abbrev	Name	Description
T_INTLIFE	Interest Lifetime (Section 3.4.1)	The time an Interest should stay pending at an intermediate node.
T_CACHETIME	Recommended Cache Time (Section 3.4.2)	The Recommended Cache Time for Content Objects.
T_MSGHASH	Message Hash (Section 3.4.3)	The hash of the CCNx Message to end of packet using Section 3.3.3 format.

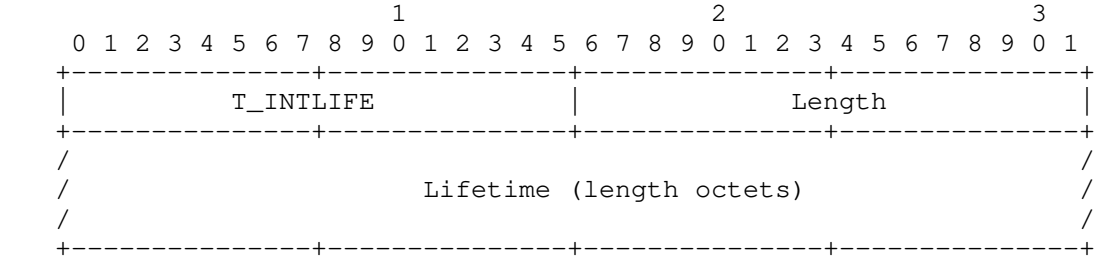
Table 4: Hop-by-hop Header Types

Additional hop-by-hop headers are defined in higher level specifications such as the fragmentation specification.

3.4.1. Interest Lifetime

The Interest Lifetime is the time that an Interest should stay pending at an intermediate node. It is expressed in milliseconds as an unsigned, network byte order integer.

A value of 0 (encoded as 1 byte %x00) indicates the Interest does not elicit a Content Object response. It should still be forwarded, but no reply is expected and a forwarder could skip creating a PIT entry.

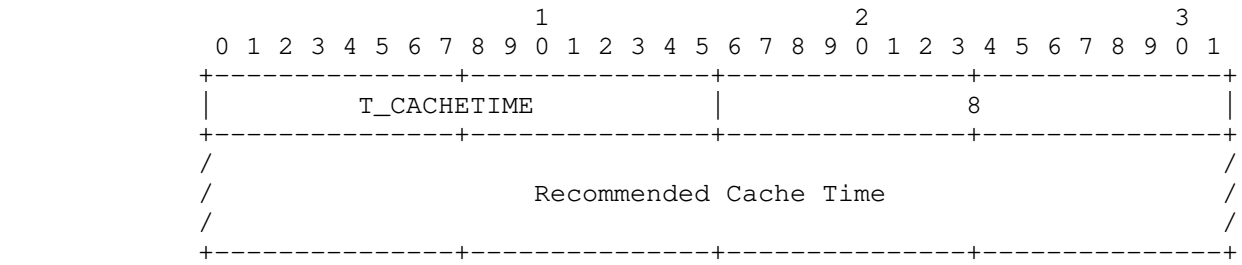


3.4.2. Recommended Cache Time

The Recommended Cache Time (RCT) is a measure of the useful lifetime of a Content Object as assigned by a content producer or upstream node. It serves as a guideline to the Content Store cache in determining how long to keep the Content Object. It is a recommendation only and may be ignored by the cache. This is in contrast to the ExpiryTime (described in Section 3.6.2.2.2) which takes precedence over the RCT and must be obeyed.

Because the Recommended Cache Time is an optional hop-by-hop header and not a part of the signed message, a content producer may re-issue a previously signed Content Object with an updated RCT without needing to re-sign the message. There is little ill effect from an attacker changing the RCT as the RCT serves as a guideline only.

The Recommended Cache Time (a millisecond timestamp) is a network byte ordered unsigned integer of the number of milliseconds since the epoch in UTC of when the payload expires. It is a 64-bit field.



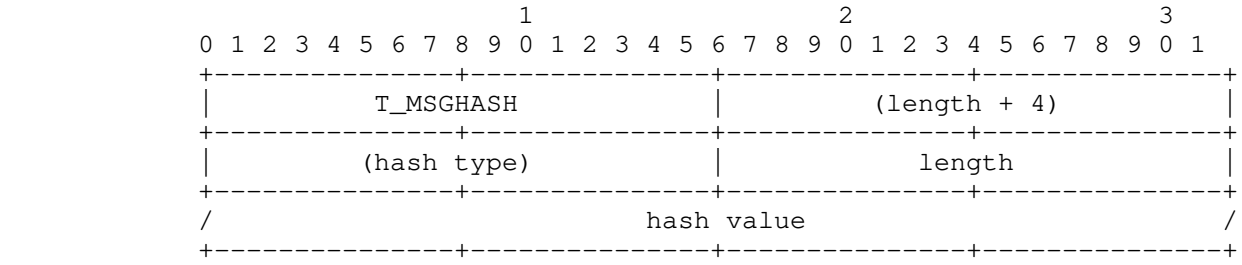
3.4.3. Message Hash

Within a trusted domain, an operator may calculate the message hash at a border device and insert that value into the hop-by-hop headers of a message. An egress device should remove the value. This permits intermediate devices within that trusted domain to match against a ContentObjectHashRestriction without calculating it at every hop.

The message hash is a cryptographic hash from the start of the CCNx Message to the end of the packet. It is used to match against the ContentObjectHashRestriction (Section 3.6.2.1.2). The Message Hash may be of longer length than an Interest's restriction, in which case the device should use the left bytes of the Message Hash to check against the Interest's value.

The Message Hash may only carry one hash type and there may only be one Message Hash header.

The Message Hash header is unprotected, so this header is only of practical use within a trusted domain, such as an operator's autonomous system.



Message Hash Header

3.5. Top-Level Types

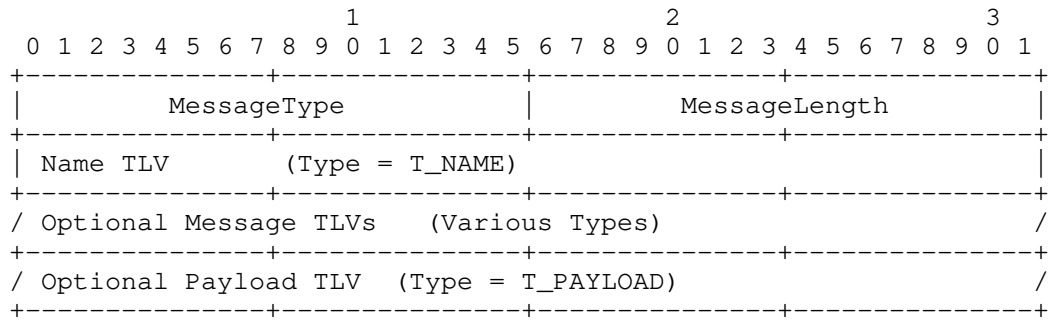
The top-level TLV types listed below exist at the outermost level of a CCNx protocol message.

Abbrev	Name	Description
T_INTEREST	Interest (Section 3.6)	An Interest MessageType.
T_OBJECT	Content Object (Section 3.6)	A Content Object MessageType
T_VALIDATION_ALG	Validation Algorithm (Section 3.6.4.1)	The method of message verification such as Message Integrity Check (MIC), a Message Authentication Code (MAC), or a cryptographic signature.
T_VALIDATION_PAYLOAD	Validation Payload (Section 3.6.4.2)	The validation output, such as the CRC32C code or the RSA signature.

Table 5: CCNx Top Level Types

3.6. CCNx Message

This is the format for the CCNx protocol message itself. The CCNx message is the portion of the packet between the hop-by-hop headers and the Validation TLVs. The figure below is an expansion of the "CCNx Message TLV" depicted in the beginning of Section 3. The CCNx message begins with MessageType and runs through the optional Payload. The same general format is used for both Interest and Content Object messages which are differentiated by the MessageType field. The first enclosed TLV of a CCNx Message is always the Name TLV. This is followed by an optional Message TLVs and an optional Payload TLV.



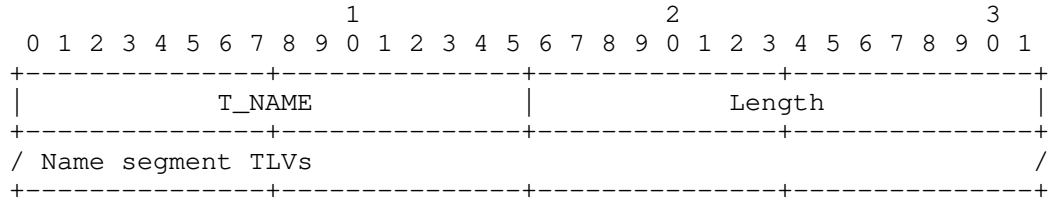
Abbrev	Name	Description
T_NAME	Name (Section 3.6.1)	The CCNx Name requested in an Interest or published in a Content Object.
T_PAYLOAD	Payload (Section 3.6.3)	The message payload.

Table 6: CCNx Message Types

3.6.1. Name

A Name is a TLV encoded sequence of segments. The table below lists the type values appropriate for these Name segments. A Name MUST NOT include PAD TLVs.

As described in CCNx Semantics [CCNSemantics], using the CCNx URI [CCNxURI] notation, a T_NAME with 0 length corresponds to ccnx:/ (the default route) and is distinct from a name with one zero length segment, such as ccnx:/NAME=. In the TLV encoding, ccnx:/ corresponds to T_NAME with 0 length, while ccnx:/NAME= corresponds to T_NAME with 4 length and T_NAMESEGMENT with 0 length.



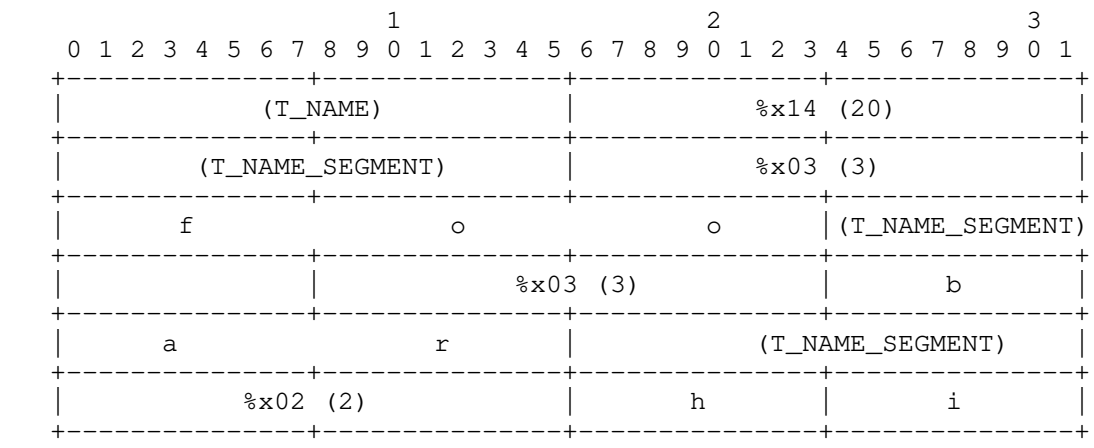
Symbolic Name	Name	Description
T_NAMESEGMENT	Name segment (Section 3.6.1.1)	A generic name Segment.
T_IPID	Interest Payload ID (Section 3.6.1.2)	An identifier that represents the Interest Payload field. As an example, the Payload ID might be a hash of the Interest Payload. This provides a way to differentiate between Interests based on their payloads without having to parse all the bytes of the payload itself; instead using only this Payload ID Name segment.
T_APP:00 - T_APP:4096	Application Components (Section 3.6.1.1)	Application-specific payload in a name segment. An application may apply its own semantics to the 4096 reserved types.

Table 7: CCNx Name Types

3.6.1.1. Name Segments

4096 special application payload name segments are allocated. These have application semantics applied to them. A good convention is to put the application's identity in the name prior to using these name segments.

For example, a name like "ccnx:/foo/bar/hi" would be encoded as:



3.6.1.2. Interest Payload ID

The InterestPayloadID is a name segment created by the origin of an Interest to represent the Interest Payload. This allows the proper multiplexing of Interests based on their name if they have different payloads. A common representation is to use a hash of the Interest Payload as the InterestPayloadID.

As part of the TLV 'value', the InterestPayloadID contains a one identifier of method used to create the InterestPayloadID followed by a variable length octet string. An implementation is not required to implement any of the methods to receive an Interest; the InterestPayloadID may be treated only as an opaque octet string for purposes of multiplexing Interests with different payloads. Only a device creating an InterestPayloadID name segment or a device verifying such a segment need to implement the algorithms.

It uses the Section 3.3.3 encoding of hash values.

In normal operations, we recommend displaying the InterestPayloadID as an opaque octet string in a CCNx URI, as this is the common denominator for implementation parsing.

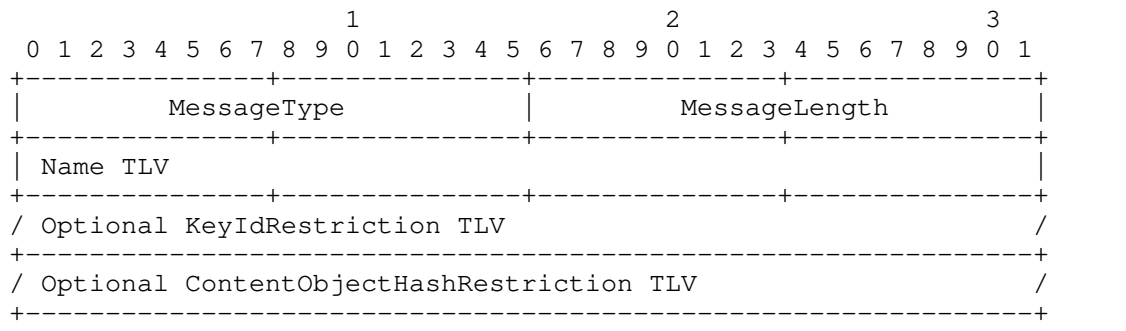
The InterestPayloadID, even if it is a hash, should not convey any security context. If a system requires confirmation that a specific entity created the InterestPayload, it should use a cryptographic signature on the Interest via the ValidationAlgorithm and ValidationPayload or use its own methods inside the Interest Payload.

3.6.2. Message TLVs

Each message type (Interest or Content Object) is associated with a set of optional Message TLVs. Additional specification documents may extend the types associated with each.

3.6.2.1. Interest Message TLVs

There are two Message TLVs currently associated with an Interest message: the KeyIdRestriction selector and the ContentObjectHashRestr selector are used to narrow the universe of acceptable Content Objects that would satisfy the Interest.



Abbrev	Name	Description
T_KEYIDRESTR	KeyIdRestriction (Section 3.6.2.1.1)	A Section 3.3.3 representation of the KeyId
T_OBJHASHRESTR	ContentObjectHashRestriction (Section 3.6.2.1.2)	A Section 3.3.3 representation of the hash of the specific Content Object that would satisfy the Interest.

Table 8: CCNx Interest Message TLV Types

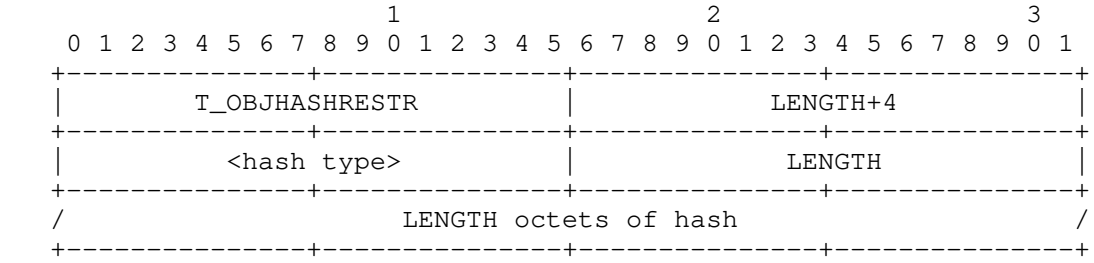
3.6.2.1.1. KeyIdRestriction

An Interest MAY include a KeyIdRestriction selector. This ensures that only Content Objects with matching KeyIds will satisfy the Interest. See Section 3.6.4.1.4.1 for the format of a KeyId.

3.6.2.1.2. ContentObjectHashRestriction

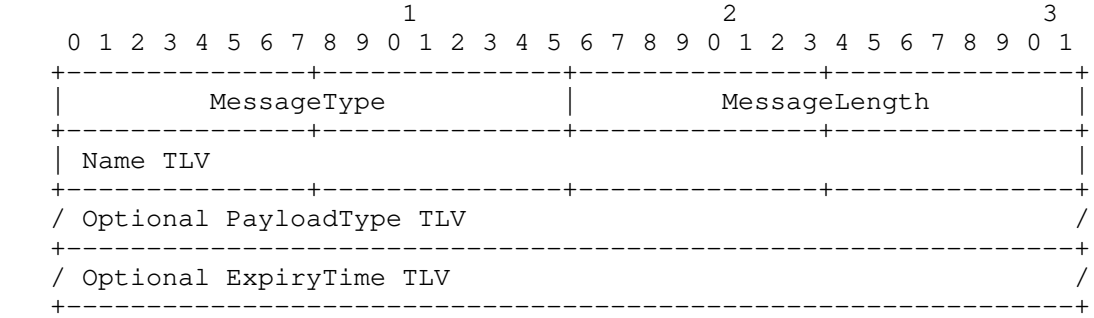
An Interest MAY contain a ContentObjectHashRestriction selector. This is the hash of the Content Object - the self-certifying name restriction that must be verified in the network, if an Interest carried this restriction. It is calculated from the beginning of the CCNx Message to the end of the packet. The LENGTH MUST be from one of the allowed values for that hash (see Section 3.3.3).

The ContentObjectHashRestriction SHOULD be of type T_SHA-256 and of length 32 bytes.



3.6.2.2. Content Object Message TLVs

The following message TLVs are currently defined for Content Objects: PayloadType (optional) and ExpiryTime (optional).



Abbrev	Name	Description
T_PAYLDTYPE	PayloadType (Section 3.6.2.2.1)	Indicates the type of Payload contents.
T_EXPIRY	ExpiryTime (Section 3.6.2.2.2)	The time at which the Payload expires, as expressed in the number of milliseconds since the epoch in UTC. If missing, Content Object may be used as long as desired.

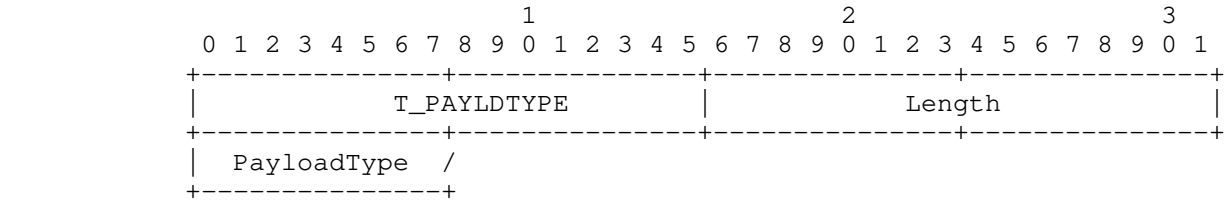
Table 9: CCNx Content Object Message TLV Types

3.6.2.2.1. PayloadType

The PayloadType is a network byte order integer representing the general type of the Payload TLV.

- T_PAYLOADTYPE_DATA: Data (possibly encrypted)
- T_PAYLOADTYPE_KEY: Key
- T_PAYLOADTYPE_LINK: Link

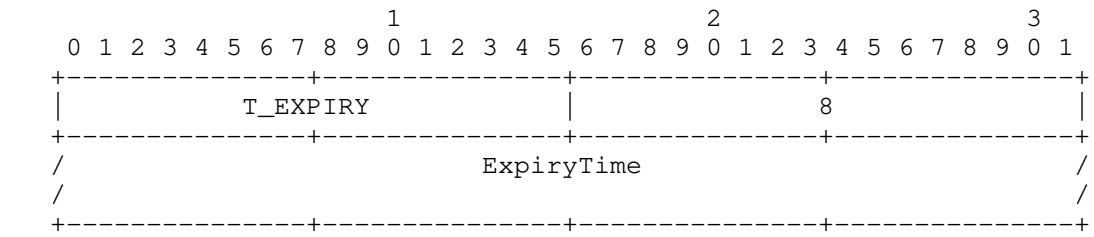
The Data type indicate that the Payload of the ContentObject is opaque application bytes. The Key type indicates that the Payload is a DER encoded public key. The Link type indicates that the Payload is one or more Link (Section 3.3.4). If this field is missing, a "Data" type is assumed.



3.6.2.2.2. ExpiryTime

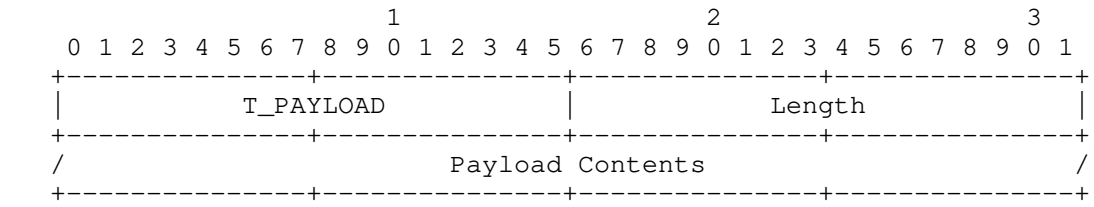
The ExpiryTime is the time at which the Payload expires, as expressed by a timestamp containing the number of milliseconds since the epoch in UTC. It is a network byte order unsigned integer in a 64-bit field. A cache or end system should not respond with a Content Object past its ExpiryTime. Routers forwarding a Content Object do

not need to check the ExpiryTime. If the ExpiryTime field is missing, the Content Object has no expressed expiration and a cache or end system may use the Content Object for as long as desired.



3.6.3. Payload

The Payload TLV contains the content of the packet. It MAY be of zero length. If a packet does not have any payload, this field MAY be omitted, rather than carrying a zero length.



3.6.4. Validation

Both Interests and Content Objects have the option to include information about how to validate the CCNx message. This information is contained in two TLVs: the ValidationAlgorithm TLV and the ValidationPayload TLV. The ValidationAlgorithm TLV specifies the mechanism to be used to verify the CCNx message. Examples include verification with a Message Integrity Check (MIC), a Message Authentication Code (MAC), or a cryptographic signature. The ValidationPayload TLV contains the validation output, such as the CRC32C code or the RSA signature.

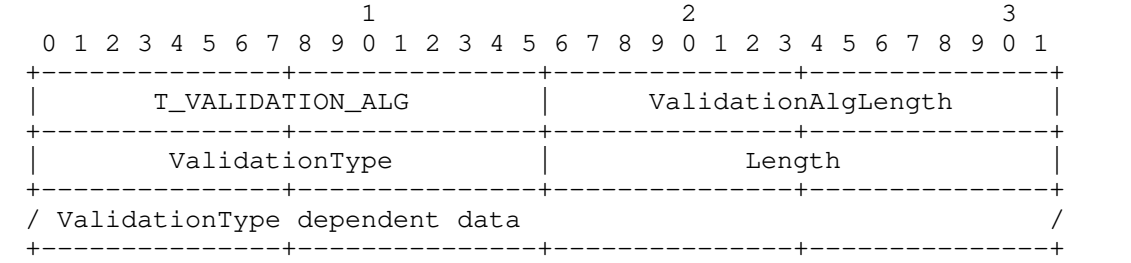
An Interest would most likely only use a MIC type of validation - a crc, checksum, or digest.

3.6.4.1. Validation Algorithm

The ValidationAlgorithm is a set of nested TLVs containing all of the information needed to verify the message. The outermost container has type = T_VALIDATION_ALG. The first nested TLV defines the specific type of validation to be performed on the message. The type

is identified with the "ValidationType" as shown in the figure below and elaborated in the table below. Nested within that container are the TLVs for any ValidationType dependent data, for example a Key Id, Key Locator etc.

Complete examples of several types may be found in Section 3.6.4.1.5



Abbrev	Name	Description
T_CRC32C	CRC32C (Section 3.6.4.1.1)	Castagnoli CRC32 (iSCSI, ext4, etc.), with normal form polynomial 0x1EDC6F41.
T_HMAC-SHA256	HMAC-SHA256 (Section 3.6.4.1.2)	HMAC (RFC 2104) using SHA256 hash.
T_RSA-SHA256	RSA-SHA256 (Section 3.6.4.1.3)	RSA public key signature using SHA256 digest.
EC-SECP-256K1	SECP-256K1 (Section 3.6.4.1.3)	Elliptic Curve signature with SECP-256K1 parameters (see [ECC]).
EC-SECP-384R1	SECP-384R1 (Section 3.6.4.1.3)	Elliptic Curve signature with SECP-384R1 parameters (see [ECC]).

Table 10: CCNx Validation Types

3.6.4.1.1. Message Integrity Checks

MICs do not require additional data in order to perform the verification. An example is CRC32C that has a "0" length value.

3.6.4.1.2. Message Authentication Checks

MACs are useful for communication between two trusting parties who have already shared private keys. Examples include an RSA signature of a SHA256 digest or others. They rely on a KeyId. Some MACs might use more than a KeyId, but those would be defined in the future.

3.6.4.1.3. Signature

Signature type Validators specify a digest mechanism and a signing algorithm to verify the message. Examples include RSA signature og a SHA256 digest, an Elliptic Curve signature with SECP-256K1 parameters, etc. These Validators require a KeyId and a mechanism for locating the publishers public key (a KeyLocator) - optionally a PublicKey or Certificate or KeyLink.

3.6.4.1.4. Validation Dependent Data

Different Validation Algorithms require access to different pieces of data contained in the ValidationAlgorithm TLV. As described above, Key Ids, Key Locators, Public Keys, Certificates, Links and Key Names all play a role in different Validation Algorithms. Any number of Validation Dependent Data containers can be present in a Validation Algorithm TLV.

Following is a table of CCNx ValidationType dependent data types:

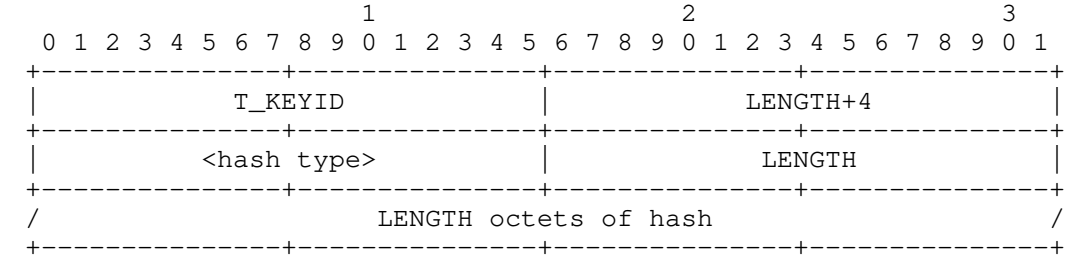
Abbrev	Name	Description
T_KEYID	SignerKeyId (Section 3.6.4.1.4.1)	An identifier of the shared secret or public key associated with a MAC or Signature.
T_PUBLICKEY	Public Key (Section 3.6.4.1.4.2)	DER encoded public key.
T_CERT	Certificate (Section 3.6.4.1.4.3)	DER encoded X509 certificate.
T_KEYLINK	KeyLink (Section 3.6.4.1.4.4)	A CCNx Link object.
T_SIGTIME	SignatureTime (Section 3.6.4.1.4.5)	A millsecond timestamp indicating the time when the signature was created.

Table 11: CCNx Validation Dependent Data Types

3.6.4.1.4.1. KeyId

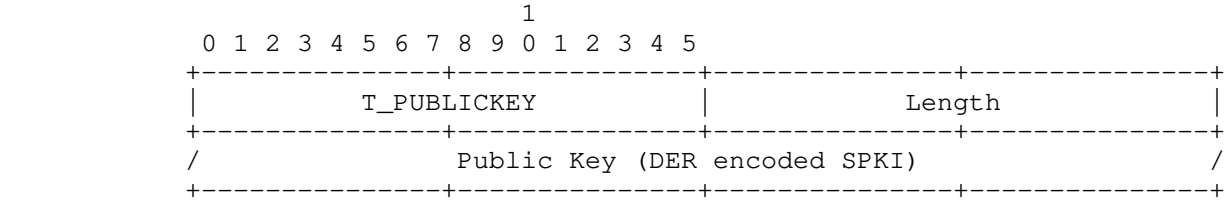
The KeyId is the publisher key identifier. It is similar to a Subject Key Identifier from X509 [RFC 5280, Section 4.2.1.2]. It should be derived from the key used to sign, such as from the SHA-256 hash of the key. It applies to both public/private key systems and to symmetric key systems.

The KeyId is represented using the Section 3.3.3. If a protocol uses a non-hash identifier, it should use one of the reserved values.

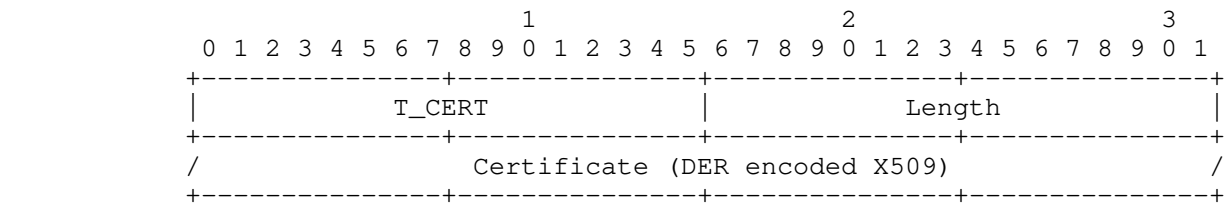


3.6.4.1.4.2. Public Key

A Public Key is a DER encoded Subject Public Key Info block, as in an X509 certificate.



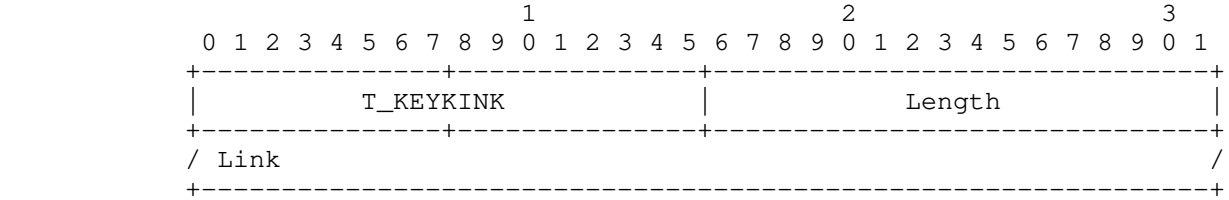
3.6.4.1.4.3. Certificate



3.6.4.1.4.4. KeyLink

A KeyLink type KeyLocator is a Link.

The KeyLink ContentObjectHashRestr, if included, is the digest of the Content Object identified by KeyLink, not the digest of the public key. Likewise, the KeyIdRestr of the KeyLink is the KeyId of the ContentObject, not necessarily of the wrapped key.

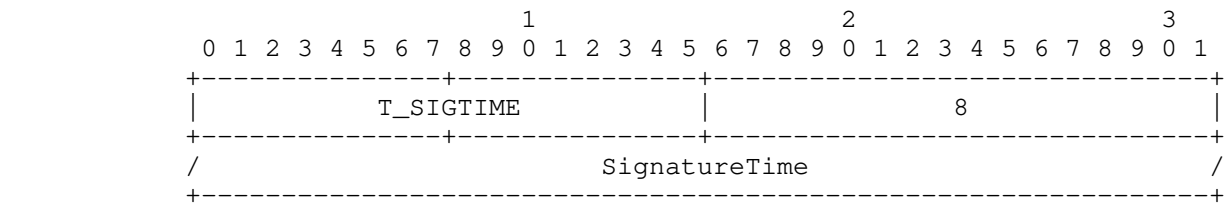


3.6.4.1.4.5. SignatureTime

The SignatureTime is a millisecond timestamp indicating the time at which a signature was created. The signer sets this field to the current time when creating a signature. A verifier may use this time to determine whether or not the signature was created during the validity period of a key, or if it occurred in a reasonable sequence with other associated signatures. The SignatureTime is unrelated to

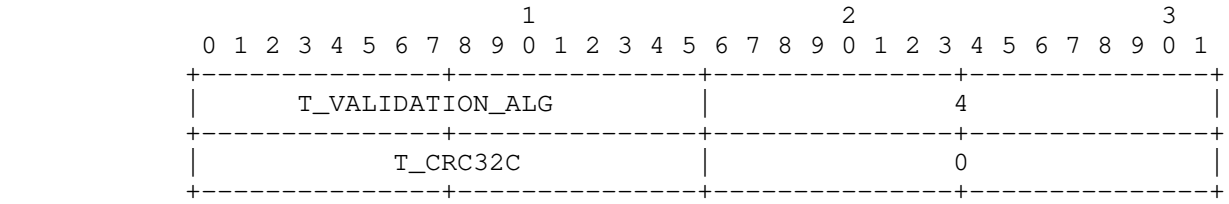
any time associated with the actual CCNx Message, which could have been created long before the signature. The default behavior is to always include a SignatureTime when creating an authenticated message (e.g. HMAC or RSA).

SignatureTime is a network byte ordered unsigned integer of the number of milliseconds since the epoch in UTC of when the signature was created. It is a fixed 64-bit field.

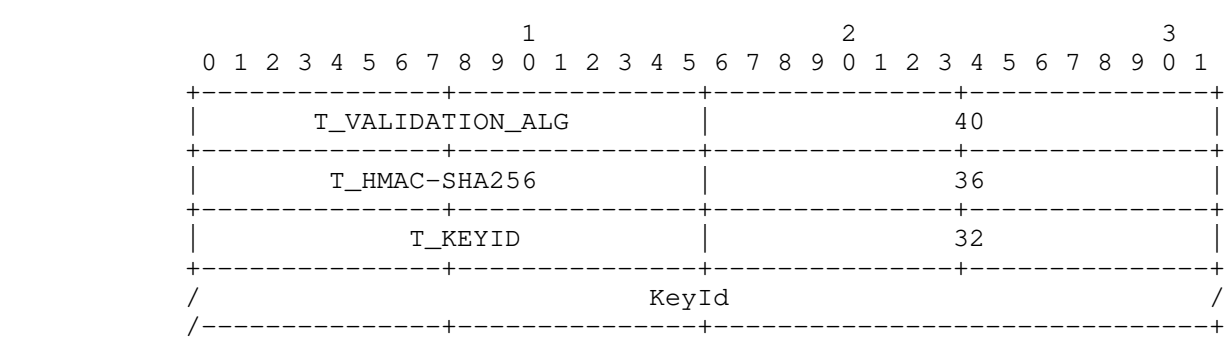


3.6.4.1.5. Validation Examples

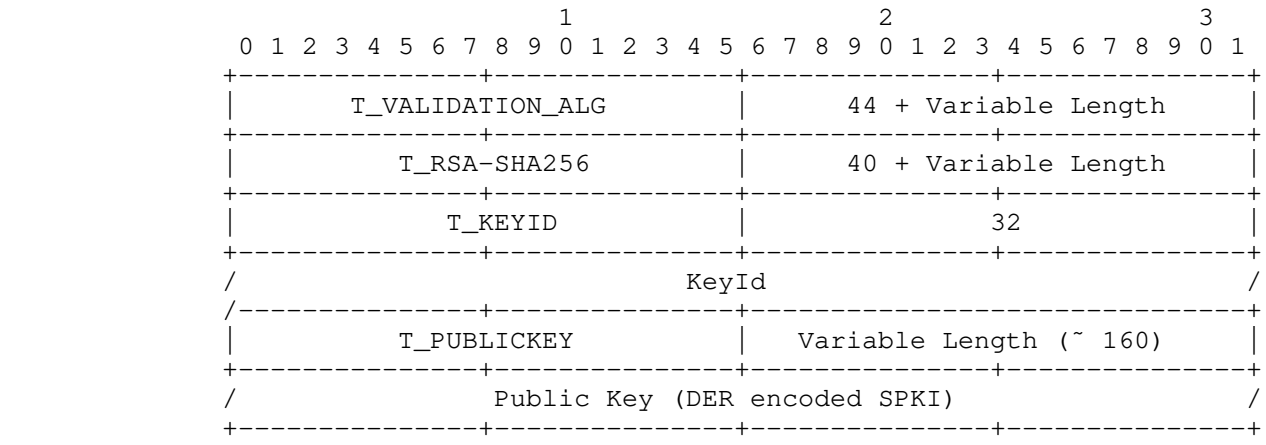
As an example of a MIC type validation, the encoding for CRC32C validation would be:



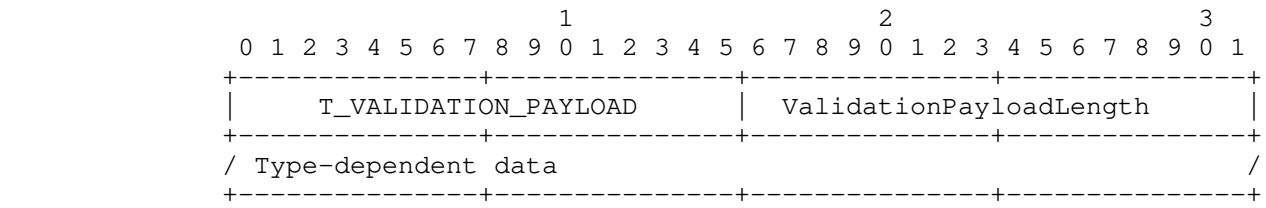
As an example of a MAC type validation, the encoding for an HMAC using a SHA256 hash would be:



As an example of a Signature type validation, the encoding for an RSA public key signing using a SHA256 digest and Public Key would be:



3.6.4.2. Validation Payload



The ValidationPayload contains the validation output, such as the CRC32C code or the RSA signature.

4. IANA Considerations

This section details each kind of protocol value that can be registered. Each type registry can be updated by incrementally expanding the type space, i.e., by allocating and reserving new types. As per [RFC5226] this section details the creation of the "CCNx Registry" and several sub-registries.

Property	Value
Name	CCNx Registry
Abbrev	CCNx

Registry Creation

4.1. Packet Type Registry

The following packet types should be allocated. A PacketType MUST be 1 byte. New packet types are allocated via "RFC Required" action.

Property	Value
Name	Packet Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	1 octet

Registry Creation

Type	Name	Reference
%x00	PT_INTEREST	Fixed Header Types (Section 3.2)
%x01	PT_CONTENT	Fixed Header Types (Section 3.2)
%x02	PT_RETURN	Fixed Header Types (Section 3.2)

Packet Type Namespace

4.2. Interest Return Code Registry

The following InterestReturn code types should be allocated.

Property	Value
Name	Interest Return Code
Parent	CCNx Registry
Review process	Specification Required
Syntax	1 octet

Registry Creation

Type	Name	Reference
%x00	Reserved	
%x01	T_RETURN_NO_ROUTE	Fixed Header Types (Section 3.2.3.3)
%x02	T_RETURN_LIMIT_EXCEEDED	Fixed Header Types (Section 3.2.3.3)
%x03	T_RETURN_NO_RESOURCES	Fixed Header Types (Section 3.2.3.3)
%x04	T_RETURN_PATH_ERROR	Fixed Header Types (Section 3.2.3.3)
%x05	T_RETURN_PROHIBITED	Fixed Header Types (Section 3.2.3.3)
%x06	T_RETURN_CONGESTED	Fixed Header Types (Section 3.2.3.3)
%x07	T_RETURN_MTU_TOO_LARGE	Fixed Header Types (Section 3.2.3.3)
%x08	T_RETURN_UNSUPPORTED_HASH_RESTRICTION	Fixed Header Types (Section 3.2.3.3)
%x09	T_RETURN_MALFORMED_INTEREST	Fixed Header Types (Section 3.2.3.3)

Interest Return Type Namespace

4.3. Hop-by-Hop Type Registry

The following hop-by-hop types should be allocated.

Property	Value
Name	Hop-by-Hop Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_INTLIFE	Hop-by-hop TLV headers (Section 3.4)
%x0002	T_CACHETIME	Hop-by-hop TLV headers (Section 3.4)
%x0003	T_MSGHASH	Hop-by-hop TLV headers (Section 3.4)
%x0004 - %x0007	Reserved	
%x0FFE	T_PAD	Pad (Section 3.3.1)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

Hop-by-Hop Type Namespace

4.4. Top-Level Type Registry

The following top-level types should be allocated.

Property	Value
Name	Top-Level Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_INTEREST	Top-Level Types (Section 3.5)
%x0002	T_OBJECT	Top-Level Types (Section 3.5)
%x0003	T_VALIDATION_ALG	Top-Level Types (Section 3.5)
%x0004	T_VALIDATION_PAYLOAD	Top-Level Types (Section 3.5)

Top-Level Type Namespace

4.5. Name Segment Type Registry

The following name segment types should be allocated.

Property	Value
Name	Name Segment Type Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_NAMESEGMENT	Name (Section 3.6.1)
%x0002	T_IPID	Name (Section 3.6.1)
%x0010 - %x0013	Reserved	Used in other drafts
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000 - %x1FFF	T_APP:00 - T_APP:4096	Application Components (Section 3.6.1)

Name Segment Type Namespace

4.6. Message Type Registry

The following CCNx message segment types should be allocated.

Property	Value
Name	Message Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	T_NAME	Message Types (Section 3.6)
%x0001	T_PAYLOAD	Message Types (Section 3.6)
%x0002	T_KEYIDRESTR	Message Types (Section 3.6)
%x0003	T_OBJHASHRESTR	Message Types (Section 3.6)
%x0005	T_PAYLDTYPE	Content Object Message Types (Section 3.6.2.2)
%x0006	T_EXPIRY	Content Object Message Types (Section 3.6.2.2)
%x0007 - %x000C	Reserved	Used in other RFC drafts
%x0FFE	T_PAD	Pad (Section 3.3.1)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

CCNx Message Type Namespace

4.7. Payload Type Registry

The following payload types should be allocated.

Property	Value
Name	PayloadType Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	Variable length unsigned integer

Registry Creation

Type	Name	Reference
%x00	T_PAYLOADTYPE_DATA	Payload Types (Section 3.6.2.2.1)
%x01	T_PAYLOADTYPE_KEY	Payload Types (Section 3.6.2.2.1)
%x02	T_PAYLOADTYPE_LINK	Payload Types (Section 3.6.2.2.1)

Payload Type Namespace

4.8. Validation Algorithm Type Registry

The following validation algorithm types should be allocated. Note: registration requires public specification of the algorithm.

Property	Value
Name	Validation Algorithm Type Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	Unassigned	
%x0002	T_CRC32C	Validation Algorithm (Section 3.6.4.1)
%x0003	Unassigned	
%x0004	T_HMAC-SHA256	Validation Algorithm (Section 3.6.4.1)
%x0005	T_RSA-SHA256	Validation Algorithm (Section 3.6.4.1)
%x0006	EC-SECP-256K1	Validation Algorithm (Section 3.6.4.1)
%x0007	EC-SECP-384R1	Validation Algorithm (Section 3.6.4.1)
%x0FFE	T_PAD	Pad (Section 3.3.1)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

Validation Algorithm Type Namespace

4.9. Validation Dependent Data Type Registry

The following validation dependent data types should be allocated.

Property	Value
Name	Validation Dependent Data Type Registry
Parent	CCNx Registry
Review process	RFC Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001 - %x0008	Unassigned	
%x0009	T_KEYID	Validation Dependent Data (Section 3.6.4.1.4)
%x000A	T_PUBLICKEYLOC	Validation Dependent Data (Section 3.6.4.1.4)
%x000B	T_PUBLICKEY	Validation Dependent Data (Section 3.6.4.1.4)
%x000C	T_CERT	Validation Dependent Data (Section 3.6.4.1.4)
%x000D	T_LINK	Validation Dependent Data (Section 3.6.4.1.4)
%x000E	T_KEYLINK	Validation Dependent Data (Section 3.6.4.1.4)
%x000F	T_SIGTIME	Validation Dependent Data (Section 3.6.4.1.4)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

Validation Dependent Data Type Namespace

4.10. Hash Function Type Registry

The following CCNx hash function types should be allocated. Note: registration requires public specification of the algorithm.

Property	Value
Name	Hash Function Type Registry
Parent	CCNx Registry
Review process	Specification Required
Syntax	2 octet TLV type

Registry Creation

Type	Name	Reference
%x0000	Reserved	
%x0001	T_SHA-256	Hash Format (Section 3.3.3)
%x0002	T_SHA-512	Hash Format (Section 3.3.3)
%x0FFF	T_ORG	Organization-Specific TLVs (Section 3.3.2)
%x1000-%x1FFF	Reserved	Experimental Use (Section 3)

CCNx Hash Function Type Namespace

5. Security Considerations

The CCNx protocol is a layer 3 network protocol, which may also operate as an overlay using other transports, such as UDP or other tunnels. It includes intrinsic support for message authentication via a signature (e.g. RSA or elliptic curve) or message authentication code (e.g. HMAC). In lieu of an authenticator, it may instead use a message integrity check (e.g. SHA or CRC). CCNx does not specify an encryption envelope, that function is left to a high-layer protocol (e.g. [esic]).

The CCNx message format includes the ability to attach MICs (e.g. SHA-256 or CRC), MACs (e.g. HMAC), and Signatures (e.g. RSA or ECDSA) to all packet types. This does not mean that it is a good idea to use an arbitrary ValidationAlgorithm, nor to include computationally expensive algorithms in Interest packets, as that could lead to computational DoS attacks. Applications should use an

explicit protocol to guide their use of packet signatures. As a general guideline, an application might use a MIC on an Interest to detect unintentionally corrupted packets. If one wishes to secure an Interest, one should consider using an encrypted wrapper and a protocol that prevents replay attacks, especially if the Interest is being used as an actuator. Simply using an authentication code or signature does not make an Interests secure. There are several examples in the literature on how to secure ICN-style messaging [mobile] [ace].

As a layer 3 protocol, this document does not describe how one arrives at keys or how one trusts keys. The CCNx content object may include a public key embedded in the object or may use the PublicKeyLocator field to point to a public key (or public key certificate) that authenticates the message. One key exchange specification is CCNxKE [ccnxke] [mobile], which is similar to the TLS 1.3 key exchange except it is over the CCNx layer 3 messages. Trust is beyond the scope of a layer-3 protocol protocol and left to applications or application frameworks.

The combination of an ephemeral key exchange (e.g. CCNxKE [ccnxke]) and an encapsulating encryption (e.g. [esic]) provides the equivalent of a TLS tunnel. Intermediate nodes may forward the Interests and Content Objects, but have no visibility inside. It also completely hides the internal names in those used by the encryption layer. This type of tunneling encryption is useful for content that has little or no cache-ability as it can only be used by someone with the ephemeral key. Short term caching may help with lossy links or mobility, but long term caching is usually not of interest.

Broadcast encryption or proxy re-encryption may be useful for content with multiple uses over time or many consumers. There is currently no recommendation for this form of encryption.

The specific encoding of messages will have security implications. This document uses a type-length-value (TLV) encoding. We chose to compromise between extensibility and unambiguous encodings of types and lengths. Some TLVs use variable length T and variable length L fields to accomodate a wide gamut of values while trying to be byte-efficient. Our TLV encoding uses a fixed length 2-byte T and 2-byte L. Using a fixed-length T and L field solves two problems. The first is aliases. If one is able to encode the same value, such as 0x2 and 0x02, in different byte lengths then one must decide if they mean the same thing, if they are different, or if one is illegal. If they are different, then one must always compare on the buffers not the integer equivalents. If one is illegal, then one must validate the TLV encoding -- every field of every packet at every hop. If they are the same, then one has the second problem: how to specify

packet filters. For example, if a name has 6 name components, then there are 7 T's and 7 L's, each of which might have up to 4 representations of the same value. That would be 14 fields with 4 encodings each, or 1001 combinations. It also means that one cannot compare, for example, a name via a memory function as one needs to consider that any embedded T or L might have a different format.

The Interest Return message has no authenticator from the previous hop. Therefore, the payload of the Interest Return should only be used locally to match an Interest. A node should never forward that Interest payload as an Interest. It should also verify that it sent the Interest in the Interest Return to that node and not allow anyone to negate Interest messages.

Caching nodes must take caution when processing content objects. It is essential that the Content Store obey the rules outlined in [CCNSemantics] to avoid certain types of attacks. Unlike NDN, CCNx 1.0 has no mechanism to work around an undesired result from the network (there are no "excludes"), so if a cache becomes poisoned with bad content it might cause problems retrieving content. There are three types of access to content from a content store: unrestricted, signature restricted, and hash restricted. If an Interest has no restrictions, then the requester is not particular about what they get back, so any matching cached object is OK. In the hash restricted case, the requester is very specific about what they want and the content store (and every forward hop) can easily verify that the content matches the request. In the signature verified case (often used for initial manifest discovery), the requester only knows the KeyId that signed the content. It is this case that requires the closest attention in the content store to avoid amplifying bad data. The content store must only respond with a content object if it can verify the signature -- this means either the content object carries the public key inside it or the Interest carries the public key in addition to the KeyId. If that is not the case, then the content store should treat the Interest as a cache miss and let an endpoint respond.

A user-level cache could perform full signature verification by fetching a public key according to the PublicKeyLocator. That is not, however, a burden we wish to impose on the forwarder. A user-level cache could also rely on out-of-band attestation, such as the cache operator only inserting content that it knows has the correct signature.

The CCNx grammar allows for hash algorithm agility via the HashType. It specifies a short list of acceptable hash algorithms that should be implemented at each forwarder. Some hash values only apply to end systems, so updating the hash algorithm does not affect forwarders --

they would simply match the buffer that includes the type-length-hash buffer. Some fields, such as the ConObjHash, must be verified at each hop, so a forwarder (or related system) must know the hash algorithm and it could cause backward compatibility problems if the hash type is updated.

A CCNx name uses binary matching whereas a URI uses a case insensitive hostname. Some systems may also use case insensitive matching of the URI path to a resource. An implication of this is that human-entered CCNx names will likely have case or non-ASCII symbol mismatches unless one uses a consistent URI normalization to the CCNx name. It also means that an entity that registers a CCNx routable prefix, say ccnx:/example.com, would need separate registrations for simple variations like ccnx:/Example.com. Unless this is addressed in URI normalization and routing protocol conventions, there could be phishing attacks.

For a more general introduction to ICN-related security concerns and approaches, see [RFC7927] and [RFC7945]

6. References

6.1. 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>.

6.2. Informative References

[ace] Shang, W., Yu, Y., Liang, T., Zhang, B., and L. Zhang, "NDN-ACE: Access control for constrained environments over named data networking", NDN Technical Report NDN-0036, 2015, <http://new.named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf>.

[CCNSemantics] Mosko, M., Solis, I., and C. Wood, "CCNx Semantics (Internet draft)", 2018, <https://www.ietf.org/id/draft-irtf-icnrg-ccnxsemantics-09.txt>.

[ccnxke] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", 2017, <https://www.ietf.org/archive/id/draft-wood-icnrg-ccnxkeyexchange-02.txt>.

[CCNxURI] Mosko, M. and C. Wood, "The CCNx URI Scheme (Internet draft)", 2017, <http://tools.ietf.org/html/draft-mosko-icnrg-ccnxuri-02>.

[CCNxz] Mosko, M., "CCNxz TLV Header Compression Experimental Code", 2016-2018, <https://github.com/PARC/CCNxz>.

[compress] Mosko, M., "Header Compression for TLV-based Packets", 2016, <https://datatracker.ietf.org/meeting/interim-2016-icnrg-02/materials/slides-interim-2016-icnrg-2-7>.

[ECC] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", 2010, <http://www.secg.org/sec2-v2.pdf>.

[EpriseNumbers] IANA, "IANA Private Enterprise Numbers", 2015, <http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>.

[esic] Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", 2017, <https://www.ietf.org/id/draft-wood-icnrg-esic-01.txt>.

[mobile] Mosko, M., Uzun, E., and C. Wood, "Mobile Sessions in Content-Centric Networks", IFIP Networking, 2017, <http://dl.ifip.org/db/conf/networking/networking2017/1570334964.pdf>.

[nnc] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", 2009, <http://dx.doi.org/10.1145/1658939.1658941>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC7927] Kutscher, D., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges", 2016, <https://trac.tools.ietf.org/html/rfc7927>.

[RFC7945] Pentikousis, K., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", 2016, <https://trac.tools.ietf.org/html/rfc7945>.

Authors' Addresses

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com

Ignacio Solis
LinkedIn
Mountain View, California 94043
USA

Email: nsolis@linkedin.com

Christopher A. Wood
University of California Irvine
Irvine, California 92697
USA

Phone: +01 315-806-5939
Email: woodc1@uci.edu

ICNRG
Internet-Draft
Intended status: Experimental
Expires: July 28, 2019

M. Mosko
PARC, Inc.
I. Solis
LinkedIn
C. Wood
University of California Irvine

January 24, 2019

CCNx Semantics
draft-irtf-icnrg-ccnxsemantics-10

Abstract

This document describes the core concepts of the Content Centric Networking (CCNx) architecture and presents a network protocol based on two messages: Interests and Content Objects. It specifies the set of mandatory and optional fields within those messages and describes their behavior and interpretation. This architecture and protocol specification is independent of a specific wire encoding.

The protocol also uses a Control message called an InterestReturn, whereby one system can return an Interest message to the previous hop due to an error condition. This indicates to the previous hop that the current system will not respond to the Interest.

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 http://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 July 28, 2019.

Table of Contents

1. Introduction

This document describes the principles of the CCNx architecture. It describes a network protocol that uses a hierarchical name to forward requests and to match responses to requests. It does not use endpoint addresses, such as Internet Protocol. Restrictions in a request can limit the response by the public key of the response's signer or the cryptographic hash of the response. Every CCNx forwarder along the path does the name matching and restriction checking. The CCNx protocol fits within the broader framework of Information Centric Networking (ICN) protocols [RFC7927]. This document concerns the semantics of the protocol and is not dependent on a specific wire format encoding. The CCNx Messages [CCNMessages] document describes a type-length-value (TLV) wire protocol encoding. This section introduces the main concepts of CCNx, which are further elaborated in the remainder of the document.

The CCNx protocol derives from the early ICN work by Jacobson et al. [nnc]. Jacobson's version of CCNx is known as the 0.x version ("CCNx 0.x") and the present work is known as the 1.0 version ("CCNx 1.0"). There are two active implementations of CCNx 1.0. The most complete implementation is Community ICN (CINC) [cicn], a Linux Foundation project hosted at fd.io. Another active implementation is CCN-lite [ccnlite], with support for IoT systems and the RIOT operating system. CCNx 0.x formed the basis of the Named Data Networking [ndn] (NDN) university project.

The current CCNx 1.0 specification diverges from CCNx 0.x in a few significant areas. The most pronounced behavioral difference between CCNx 0.x and CCNx 1.0 is that CCNx 1.0 has a simpler response processing behavior. In both versions, a forwarder uses a hierarchical longest prefix match of a request name against the forwarding information base (FIB) to send the request through the network to a system that can issue a response. A forwarder must then match a response's name to a request's name to determine the reverse

path and deliver the response to the requester. In CCNx 0.x, the Interest name may be a hierarchical prefix of the response name, which allows a form of layer 3 content discovery. In CCNx 1.0, a response's name must exactly equal a request's name. Content discovery is performed by a higher-layer protocol.

CCNx Selectors [selectors] is an example of using a higher-layer protocol on top of the CCNx 1.0 layer-3 to perform content discovery. The selector protocol uses a method similar to the original CCNx 0.x techniques without requiring partial name matching of a response to a request in the forwarder.

The document represents the consensus of the ICN RG. It is the first ICN protocol from the RG, created from the early CCNx protocol [nnc] with significant revision and input from the ICN community and RG members. The draft has received critical reading by several members of the ICN community and the RG. The authors and RG chairs approve of the contents. The document is sponsored under the IRTF and is not issued by the IETF and is not an IETF standard. This is an experimental protocol and may not be suitable for any specific application and the specification may change in the future.

1.1. 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].

1.2. Architecture

We describe the architecture of the network in which CCNx operates and introduce certain terminology from [terminology]. The detailed behavior of each component and message grammars are in Section 2.

A producer (also called a publisher) is an endpoint that encapsualtes content in Content Objects for transport in the CCNx network. A producer has a public/private keypair and signs (directly or indirectly) the content objects. Usually, the producer's keyid (hash of the public key) is well-known or may be derived from the producer's namespace via standard means.

A producer operates within one or more namespaces. A namespace is a name prefix that is represented in the forwarding information base (FIB). This allows a request to reach the producer and fetch a response (if one exists).

The forwarding information base (FIB) is a table that tells a forwarder where to send a request. It may point to a local

application, a local cache or content store, or to a remote system. If there is no matching entry in the FIB, a forwarder cannot process a request. The detailed rules on name matching to the FIB are given in Section 2.4.4. An endpoint has a FIB, though it may be a simple default route. An intermediate system (i.e. a router) typically has a much larger FIB. A core CCNx forwarder, for example, would know all the global routes.

A consumer is an endpoint that requests a name. It is beyond the scope of this document to describe how a consumer learns of a name or publisher keyid -- higher layer protocols build on top of CCNx handle those tasks, such as search engines or lookup services or well known names. The consumer constructs a request, called an Interest, and forwards it via the endpoint's FIB. The consumer should get back either a response, called a Content Object, that matches the Interest or a control message, called an InterestReturn, that indicates the network cannot handle the request.

There are three ways to detect errors in Interest handling. An InterestReturn is a network control message that indicates a low-level error like no route or out of resources. If an Interest arrives at a producer, but the producer does not have the requested content, the producer should send an application-specific error message (e.g. a not found message). Finally, a consumer may not receive anything, in which case it should timeout and, depending on the application, retry the request or return an error to the application.

1.3. Protocol Overview

The goal of CCNx is to name content and retrieve the content from the network without binding it to a specific network endpoint. A routing system (specified separately) populates the forwarding information base (FIB) tables at each CCNx router with hierarchical name prefixes that point towards the content producers under that prefix. A request finds matching content along those paths, in which case a response carries the data, or if no match is found a control message indicates the failure. A request may further refine acceptable responses with a restriction on the response's signer and the cryptographic hash of the response. The details of these restrictions are described below.

The CCNx name is a hierarchical series of path segments. Each path segment has a type and zero or more bytes. Matching two names is done as a binary comparison of the type and value, segment by segment. The human-readable form is defined under a URI scheme "ccnx:" [CCNxURI], though the canonical encoding of a name is a series of (type, octet string) pairs. There is no requirement that

any path segment be human readable or UTF-8. The first few segments in a name will matched against the FIB and a routing protocol may put its own restrictions on the routable name components (e.g. a maximum length or character encoding rules). In principle, path segments and names have unbounded length, though in practice they are limited by the wire format encoding and practical considerations imposed by a routing protocol. Note that in CCNx path segments use binary comparison whereas in a URI the authority uses case-insensitive hostname (due to DNS).

The CCNx name, as used by the forwarder, is purposefully left as a general octet-encoded type and value without any requirements on human readability and character encoding. The reason for this is that we are concerned with how a forwarder processes names. We expect that applications, routing protocols, or other higher layers will apply their own conventions and restrictions on the allowed path segment types and path segment values.

CCNx is a request and response protocol to fetch chunks of data using a name. The integrity of each chunk may be directly asserted through a digital signature or Message Authentication Code (MAC), or, alternatively, indirectly via hash chains. Chunks may also carry weaker message integrity checks (MICs) or no integrity protection mechanism at all. Because provenance information is carried with each chunk (or larger indirectly protected block), we no longer need to rely on host identities, such as those derived from TLS certificates, to ascertain the chunk legitimacy. Data integrity is therefore a core feature of CCNx; it does not rely on the data transmission channel. There are several options for data confidentiality, discussed later.

This document only defines the general properties of CCNx names. In some isolated environments, CCNx users may be able to use any name they choose and either inject that name (or prefix) into a routing protocol or use other information foraging techniques. In the Internet environment, there will be policies around the formats of names and assignments of names to publishers, though those are not specified here.

The key concept of CCNx is that a subjective name is cryptographically bound to a fixed payload. These publisher-generated bindings can therefore be cryptographically verified. A named payload is thus the tuple {{Name, ExtraFields, Payload, ValidationAlgorithm}, ValidationPayload}, where all fields in the inner tuple are covered by the validation payload (e.g. signature). Consumers of this data can check the binding integrity by re-computing the same cryptographic hash and verifying the digital signature in ValidationPayload.

In addition to digital signatures (e.g. RSA), CCNx also supports message authentication codes (e.g. HMAC) and message integrity codes (e.g. SHA-256 or CRC). To maintain the cryptographic binding, there should be at least one object with a signature or authentication code, but not all objects require it. For example, a first object with a signature could refer to other objects via a hash chain, a Merkle tree, or a signed manifest. The later objects may not have any validation and rely purely on the references. The use of an integrity code (e.g. CRC) is intended for detecting accidental corruption in an Interest.

CCNx specifies a network protocol around Interests (request messages) and Content Objects (response messages) to move named payloads. An Interest includes the Name -- which identifies the desired response -- and optional matching restrictions. Restrictions limit the possible matching Content Objects. Two restrictions exist: KeyIdRestr and ContentObjectHashRestr. The first restriction on the KeyId limits responses to those signed with a ValidationAlgorithm KeyId field equal to the restriction. The second is the Content ObjectHash restriction, which limits the response to one where the cryptographic hash of the entire named payload is equal to the restriction.

The hierarchy of a CCNx Name is used for routing via the longest matching prefix in a Forwarder. The longest matching prefix is computed name segment by name segment in the hierarchical path name, where each name segment must be exactly equal to match. There is no requirement that the prefix be globally routable. Within a deployment any local routing may be used, even one that only uses a single flat (non-hierarchical) name segment.

Another concept of CCNx is that there should be flow balance between Interest messages and Content Object messages. At the network level, an Interest traveling along a single path should elicit no more than one Content Object response. If some node sends the Interest along more than one path, that node should consolidate the responses such that only one Content Object flows back towards the requester. If an Interest is sent broadcast or multicast on a multiple-access media, the sender should be prepared for multiple responses unless some other media-dependent mechanism like gossip suppression or leader election is used.

As an Interest travels the forward path following the Forwarding Information Base (FIB), it establishes state at each forwarder such that a Content Object response can trace its way back to the original requester(s) without the requester needing to include a routable return address. We use the notional Pending Interest Table (PIT) as

a method to store state that facilitates the return of a Content Object.

The notional PIT table stores the last hop of an Interest plus its Name and optional restrictions. This is the data required to match a Content Object to an Interest (see Section 9). When a Content Object arrives, it must be matched against the PIT to determine which entries it satisfies. For each such entry, at most one copy of the Content Object is sent to each listed last hop in the PIT entries.

An actual PIT table is not mandated by the specification. An implementation may use any technique that gives the same external behavior. There are, for example, research papers that use techniques like label switching in some parts of the network to reduce the per-node state incurred by the PIT table [dart]. Some implementations store the PIT state in the FIB, so there is not a second table.

If multiple Interests with the same {Name, KeyIdRestr, ContentObjectHashRestr} tuple arrive at a node before a Content Object matching the first Interest comes back, they are grouped in the same PIT entry and their last hops aggregated (see Section 2.4.2). Thus, one Content Object might satisfy multiple pending Interests in a PIT.

In CCNx, higher-layer protocols are often called "name-based protocols" because they operate on the CCNx Name. For example, a versioning protocol might append additional name segments to convey state about the version of payload. A content discovery protocol might append certain protocol-specific name segments to a prefix to discover content under that prefix. Many such protocols may exist and apply their own rules to Names. They may be layered with each protocol encapsulating (to the left) a higher layer's Name prefix.

This document also describes a control message called an InterestReturn. A network element may return an Interest message to a previous hop if there is an error processing the Interest. The returned Interest may be further processed at the previous hop or returned towards the Interest origin. When a node returns an Interest it indicates that the previous hop should not expect a response from that node for the Interest, i.e., there is no PIT entry left at the returning node for a Content Object to follow.

There are multiple ways to describe larger objects in CCNx. Aggregating layer-3 content objects in to larger objects is beyond the scope of this document. One proposed method, FLIC [flic], uses a manifest to enumerate the pieces of a larger object. Manifests are, themselves, Content Objects. Another option is to use a convention

in the Content Object name, as in the CCNx Chunking [chunking] protocol where a large object is broken in to small chunks and each chunk receives a special name component indicating its serial order.

At the semantic level, described in this document, we do not address fragmentation. One experimental fragmentation protocol, BeginEnd Fragments [befrags] uses a multipoint-PPP style technique for use over layer-2 interfaces with the CCNx Messages [CCNMessages] TLV wire forman specification.

With these concepts, the remainder of the document specifies the behavior of a forwarder in processing Interest, Content Object, and InterestReturn messages.

2. Protocol

CCNx is a request and response protocol. A request is called an Interest and a response is called a Content Object. CCNx also uses a 1-hop control message called InterestReturn. These are, as a group, called CCNx Messages.

2.1. Message Grammar

The CCNx message ABNF [RFC5234] grammar is shown in Figure 1. The grammar does not include any encoding delimiters, such as TLVs. Specific wire encodings are given in a separate document. If a Validation section exists, the Validation Algorithm covers from the Body (BodyName or BodyOptName) through the end of the ValidationAlg section. The InterestLifetime, CacheTime, and Return Code fields exist outside of the validation envelope and may be modified.

The various fields -- in alphabetical order -- are defined as:

- AbsTime: Absolute times are conveyed as the 64-bit UTC time in milliseconds since the epoch (standard POSIX time).

- CacheTime: The absolute time after which the publisher believes there is low value in caching the content object. This is a recommendation to caches (see Section 4).

- ConObjField: These are optional fields that may appear in a Content Object.

- ConObjHash: The value of the Content Object Hash, which is the SHA256-32 over the message from the beginning of the body to the end of the message. Note that this coverage area is different from the ValidationAlg. This value SHOULD NOT be trusted across domains (see Section 5).

- ExpiryTime: An absolute time after which the content object should be considered expired (see Section 4).

- Hash: Hash values carried in a Message carry a HashType to identify the algorithm used to generate the hash followed by the hash value. This form is to allow hash agility. Some fields may mandate a specific HashType.

- HopLimit: Interest messages may loop if there are loops in the forwarding plane. To eventually terminate loops, each Interest carries a HopLimit that is decremented after each hop and no longer forwarded when it reaches zero. See Section 2.4.

- InterestField: These are optional fields that may appear in an Interest message.

- KeyIdRestr: The KeyId Restriction. A Content Object must have a KeyId with the same value as the restriction.

- ContentObjectHashRestr: The Content Object Hash Restriction. A content object must hash to the same value as the restriction using the same HashType. The ContentObjectHashRestr MUST use SHA256-32.

- KeyId: An identifier for the key used in the ValidationAlg. For public key systems, this should be the SHA-256 hash of the public key. For symmetric key systems, it should be an identifier agreed upon by the parties.

- KeyLink: A Link (see Section 6) that names how to retrieve the key used to verify the ValidationPayload. A message SHOULD NOT have both a KeyLink and a PublicKey.

- Lifetime: The approximate time during which a requester is willing to wait for a response, usually measured in seconds. It is not strongly related to the network round trip time, though it must necessarily be larger.

- Name: A name is made up of a non-empty first segment followed by zero or more additional segments, which may be of 0 length. Path segments are opaque octet strings, and are thus case-sensitive if encoding UTF-8. An Interest MUST have a Name. A Content Object MAY have a Name (see Section 9). The segments of a name are said to be complete if its segments uniquely identify a single Content Object. A name is exact if its segments are complete. An Interest carrying a full name is one which specifies an exact name and the ContentObjectHashRestr of the corresponding Content Object.

- Payload: The message's data, as defined by PayloadType.

- PayloadType: The format of the Payload. If missing, assume DataType. DataType means the payload is opaque application bytes. KeyType means the payload is a DER-encoded public key. LinkType means it is one or more Links (see Section 6).

- PublicKey: Some applications may wish to embed the public key used to verify the signature within the message itself. The PublickKey is DER encoded. A message SHOULD NOT have both a KeyLink and a PublicKey.

- RelTime: A relative time, measured in milli-seconds.

- ReturnCode: States the reason an Interest message is being returned to the previous hop (see Section 10.2).

- SigTime: The absolute time (UTC milliseconds) when the signature was generated.

- Vendor: Vendor-specific opaque data. The Vendor data includes the IANA Private Enterprise Numbers [EpriseNumbers], followed by vendor-specific information. CCNx allows vendor-specific data in most locations of the grammar.

```
Message       := Interest / ContentObject / InterestReturn
Interest      := IntHdr BodyName [Validation]
IntHdr        := HopLimit [Lifetime] *Vendor
ContentObject := ConObjHdr BodyOptName [Validation]
ConObjHdr     := [CacheTime / ConObjHash] *Vendor
InterestReturn:= ReturnCode Interest
BodyName      := Name Common
BodyOptName   := [Name] Common
Common        := *Field [Payload]
Validation    := ValidationAlg ValidatonPayload

Name          := FirstSegment *Segment
FirstSegment  := 1* OCTET / Vendor
Segment       := 0* OCTET / Vendor

ValidationAlg := (RSA-SHA256 / HMAC-SHA256 / CRC32C) *Vendor
ValidatonPayload := 1* OCTET
RSA-SHA256    := KeyId [PublicKey] [SigTime] [KeyLink]
HMAC-SHA256   := KeyId [SigTime] [KeyLink]
CRC32C        := [SigTime]

AbsTime       := 8 OCTET ; 64-bit UTC msec since epoch
CacheTime     := AbsTime
```

```
ConObjField   := ExpiryTime / PayloadType
ConObjHash    := Hash ; The Content Object Hash
DataType      := "1"
ExpiryTime    := AbsTime
Field         := InterestField / ConObjField / Vendor
Hash          := HashType 1* OCTET
HashType      := SHA256-32 / SHA512-64 / SHA512-32
HopLimit      := OCTET
InterestField := KeyIdRestr / ContentObjectHashRestr
KeyId         := 1* OCTET ; key identifier
KeyIdRestr    := 1* OCTET
KeyLink       := Link
KeyType       := "2"
Lifetime      := RelTime
Link          := Name [KeyIdResr] [ContentObjectHashRestr]
LinkType      := "3"
ContentObjectHashRestr  := Hash
Payload       := *OCTET
PayloadType   := DataType / KeyType / LinkType
PublicKey     := ; DER-encoded public key
RelTime       := 1* OCTET ; msec
ReturnCode    := ; see Section 10.2
SigTime       := AbsTime
Vendor        := PEN 0* OCTET
PEN           := ; IANA Private Enterprise Number
```

Figure 1

2.2. Consumer Behavior

To request a piece of content for a given {Name, [KeyIdRest], [ContentObjectHashRestr]} tuple, a consumer creates an Interest message with those values. It MAY add a validation section, typically only a CRC32C. A consumer MAY put a Payload field in an Interest to send additional data to the producer beyond what is in the Name. The Name is used for routing and may be remembered at each hop in the notional PIT table to facilitate returning a content object; Storing large amounts of state in the Name could lead to high memory requirements. Because the Payload is not considered when forwarding an Interest or matching a Content Object to an Interest, a consumer SHOULD put an Interest Payload ID (see Section 3.2) as part of the name to allow a forwarder to match Interests to content objects and avoid aggregating Interests with different payloads. Similarly, if a consumer uses a MAC or a signature, it SHOULD also include a unique segment as part of the name to prevent the Interest from being aggregated with other Interests or satisfied by a Content Object that has no relation to the validation.

The consumer SHOULD specify an InterestLifetime, which is the length of time the consumer is willing to wait for a response. The InterestLifetime is an application-scale time, not a network round trip time (see Section 2.4.2). If not present, the InterestLifetime will use a default value (2 seconds).

The consumer SHOULD set the Interest HopLimit to a reasonable value or use the default 255. If the consumer knows the distances to the producer via routing, it SHOULD use that value.

A consumer hands off the Interest to its first forwarder, which will then forward the Interest over the network to a publisher (or replica) that may satisfy it based on the name (see Section 2.4).

Interest messages are unreliable. A consumer SHOULD run a transport protocol that will retry the Interest if it goes unanswered, up to the InterestLifetime. No transport protocol is specified in this document.

The network MAY send to the consumer an InterestReturn message that indicates the network cannot fulfill the Interest. The ReturnCode specifies the reason for the failure, such as no route or congestion. Depending on the ReturnCode, the consumer MAY retry the Interest or MAY return an error to the requesting application.

If the content was found and returned by the first forwarder, the consumer will receive a Content Object. The consumer SHOULD:

- Ensure the content object is properly formatted.
- Verify that the returned Name matches a pending request. If the request also had KeyIdRestr or ObjHashRest, it MUST also validate those properties.
- If the content object is signed, it SHOULD cryptographically verify the signature. If it does not have the corresponding key, it SHOULD fetch the key, such as from a key resolution service or via the KeyLink.
- If the signature has a SigTime, the consumer MAY use that in considering if the signature is valid. For example, if the consumer is asking for dynamically generated content, it should expect the SigTime to not be before the time the Interest was generated.
- If the content object is signed, it should assert the trustworthiness of the signing key to the namespace. Such an assertion is beyond the scope of this document, though one may use

traditional PKI methods, a trusted key resolution service, or methods like [trust].

- It MAY cache the content object for future use, up to the ExpiryTime if present.

- A consumer MAY accept a content object off the wire that is expired. It may happen that a packet expires while in flight, and there is no requirement that forwarders drop expired packets in flight. The only requirement is that content stores, caches, or producers MUST NOT respond with an expired content object.

2.3. Publisher Behavior

This document does not specify the method by which names populate a Forwarding Information Base (FIB) table at forwarders (see Section 2.4). A publisher is either configured with one or more name prefixes under which it may create content, or it chooses its name prefixes and informs the routing layer to advertise those prefixes.

When a publisher receives an Interest, it SHOULD:

- Verify that the Interest is part of the publishers namespace(s).

- If the Interest has a Validation section, verify the ValidationPayload. Usually an Interest will only have a CRC32C unless the publisher application specifically accommodates other validations. The publisher MAY choose to drop Interests that carry a Validation section if the publisher application does not expect those signatures as this could be a form of computational denial of service. If the signature requires a key that the publisher does not have, it is NOT RECOMMENDED that the publisher fetch the key over the network, unless it is part of the application's expected behavior.

- Retrieve or generate the requested content object and return it to the Interest's previous hop. If the requested content cannot be returned, the publisher SHOULD reply with an InterestReturn or a content object with application payload that says the content is not available; this content object should have a short ExpiryTime in the future or not be cacheable (i.e. an expiry time of 0).

2.4. Forwarder Behavior

A forwarder routes Interest messages based on a Forwarding Information Base (FIB), returns Content Objects that match Interests to the Interest's previous hop, and processes InterestReturn control messages. It may also keep a cache of Content Objects in the

notional Content Store table. This document does not specify the internal behavior of a forwarder -- only these and other external behaviors.

In this document, we will use two processing pipelines, one for Interests and one for Content Objects. Interest processing is made up of checking for duplicate Interests in the PIT (see Section 2.4.2), checking for a cached Content Object in the Content Store (see Section 2.4.3), and forwarding an Interest via the FIB. Content Store processing is made up of checking for matching Interests in the PIT and forwarding to those previous hops.

2.4.1. Interest HopLimit

Interest looping is not prevented in CCNx. An Interest traversing loops is eventually discarded using the hop-limit field of the Interest, which is decremented at each hop traversed by the Interest.

A loop may also terminate because the Interest is aggregated with it's previous PIT entry along the loop. In this case, the Content will be sent back along the loop and eventually return to a node that already forwarded the content, so it will likely not have a PIT entry any more. When the content reaches a node without a PIT entry, it will be discarded. It may be that a new Interest or another looped Interest will return to that same node, in which case the node will either return a cached response to make a new PIT entry, as below.

The HopLimit is the last resort method to stop Interest loops where a Content Object chases an Interest around a loop and where the intermediate nodes, for whatever reason, no longer have a PIT entry and do not cache the Content Object.

Every Interest MUST carry a HopLimit. An Interest received from a local application MAY have a 0 HopLimit, which restricts the Interest to other local sources.

When an Interest is received from another forwarder, the HopLimit MUST be positive, otherwise the forwarder will discard the Interest. A forwarder MUST decrement the HopLimit of an Interest by at least 1 before it is forwarded.

If the decremented HopLimit equals 0, the Interest MUST NOT be forwarded to another forwarder; it MAY be sent to a local publisher application or serviced from a local Content Store.

A RECOMMENDED HopLimit processing pipeline is below:

- If Interest received from a remote system:

    * If received HopLimit is 0, optionally send InterestReturn (HopLimit Exceeded), and discard Interest.

    * Otherwise, decrement the HopLimit by 1.

- Process as per Content Store and Aggregation rules.

- If the Interest will be forwarded:

    * If the (potentailly decremented) HopLimit is 0, restrict forwarding to the local system.

    * Otherwise, forward as desired to local or remote systems.

2.4.2. Interest Aggregation

Interest aggregation is when a forwarder receives an Interest message that could be satisfied by the response to another Interest message already forwarded by the node, so the forwarder suppresses forwarding the new Interest; it only records the additional previous hop so a Content Object sent in response to the first Interest will satisfy both Interests.

CCNx uses an Interest aggregation rule that assumes the InterestLifetime is akin to a subscription time and is not a network round trip time. Some previous aggregation rules assumed the lifetime was a round trip time, but this leads to problems of expiring an Interest before a response comes if the RTT is estimated too short or interfering with an ARQ scheme that wants to re-transmit an Interest but a prior interest over-estimated the RTT.

A forwarder MAY implement an Interest aggregation scheme. If it does not, then it will forward all Interest messages. This does not imply that multiple, possibly identical, Content Objects will come back. A forwarder MUST still satisfy all pending Interests, so one Content Object could satisfy multiple similar interests, even if the forwarded did not suppress duplicate Interest messages.

A RECOMMENDED Interest aggregation scheme is:

- Two Interests are considered 'similar' if they have the same Name, KeyIdRestr, and ContentObjectHashRestr.

- Let the notional value InterestExpiry (a local value at the forwarder) be equal to the receive time plus the InterestLifetime (or a platform-dependent default value if not present).

- An Interest record (PIT entry) is considered invalid if its InterestExpiry time is in the past.

- The first reception of an Interest MUST be forwarded.

- A second or later reception of an Interest similar to a valid pending Interest from the same previous hop MUST be forwarded. We consider these a retransmission requests.

- A second or later reception of an Interest similar to a valid pending Interest from a new previous hop MAY be aggregated (not forwarded). If this Interest has a larger HopLimit than the pending Interest, it MUST be forwarded.

- Aggregating an Interest MUST extend the InterestExpiry time of the Interest record. An implementation MAY keep a single InterestExpiry time for all previous hops or MAY keep the InterestExpiry time per previous hop. In the first case, the forwarder might send a Content Object down a path that is no longer waiting for it, in which case the previous hop (next hop of the Content Object) would drop it.

2.4.3. Content Store Behavior

The Content Store is a special cache that is an integral part of a CCNx forwarder. It is an optional component. It serves to repair lost packets and handle flash requests for popular content. It could be pre-populated or use opportunistic caching. Because the Content Store could serve to amplify an attack via cache poisoning, there are special rules about how a Content Store behaves.

1. A forwarder MAY implement a Content Store. If it does, the Content Store matches a Content Object to an Interest via the normal matching rules (see Section 9).

2. If an Interest has a KeyIdRestr, then the Content Store MUST NOT reply unless it knows the signature on the matching Content Object is correct. It may do this by external knowledge (i.e., in a managed network or system with pre-populated caches) or by having the public key and cryptographically verifying the signature. A Content Store is NOT REQURIED to verify signatures; if it does not, then it treats these cases like a cache miss.

3. If a Content Store chooses to verify signatures, then it MAY do so as follows. If the public key is provided in the Content Object itself (i.e., in the PublicKey field) or in the Interest, the Content Store MUST verify that the public key's SHA-256 hash is equal to the KeyId and that it verifies the signature. A

Content Store MAY verify the digital signature of a Content Object before it is cached, but it is not required to do so. A Content Store SHOULD NOT fetch keys over the network. If it cannot or has not yet verified the signature, it should treat the Interest as a cache miss.

4. If an Interest has an ContentObjectHashRestr, then the Content Store MUST NOT reply unless it knows the the matching Content Object has the correct hash. If it cannot verify the hash, then it should treat the Interest as a cache miss.

5. It must obey the Cache Control directives (see Section 4).

2.4.4. Interest Pipeline

1. Perform the HopLimit check (see Section 2.4.1).

2. Determine if the Interest can be aggregated, as per Section 2.4.2. If it can be, aggregate and do not forward the Interest.

3. If forwarding the Interest, check for a hit in the Content Store, as per Section 2.4.3. If a matching Content Object is found, return it to the Interest's previous hop. This injects the Content Store as per Section 2.4.5.

4. Lookup the Interest in the FIB. Longest prefix match (LPM) is performed name segment by name segment (not byte or bit). It SHOULD exclude the Interest's previous hop. If a match is found, forward the Interest. If no match is found or the forwarder choses to not forward due to a local condition (e.g., congestion), it SHOULD send an InterestReturn message, as per Section 10.

2.4.5. Content Object Pipeline

1. It is RECOMMENDED that a forwarder that receives a content object check that the Content Object came from an expected previous hop. An expected previous hop is one pointed to by the FIB or one recorded in the PIT as having had a matching Interest sent that way.

2. A Content Object MUST be matched to all pending Interests that satisfy the matching rules (see Section 9). Each satisfied pending Interest MUST then be removed from the set of pending Interests.

3. A forwarder SHOULD NOT send more then one copy of the received Content Object to the same Interest previous hop. It may happen, for example, that two Interest ask for the same Content Object in different ways (e.g., by name and by name an KeyId) and that they both come from the same previous hop. It is normal to send the same content object multiple times on the same interface, such as Ethernet, if it is going to different previous hops.

4. A Content Object SHOULD only be put in the Content Store if it satisfied an Interest (and passed rule #1 above). This is to reduce the chances of cache poisoning.

3. Names

A CCNx name is a composition of name segments. Each name segment carries a label identifying the purpose of the name segment, and a value. For example, some name segments are general names and some serve specific purposes, such as carrying version information or the sequencing of many chunks of a large object into smaller, signed Content Objects.

There are three different types of names in CCNx: prefix, exact, and full names. A prefix name is simply a name that does not uniquely identify a single Content Object, but rather a namespace or prefix of an existing Content Object name. An exact name is one which uniquely identifies the name of a Content Object. A full name is one which is exact and is accompanied by an explicit or implicit ConObjHash. The ConObjHash is explicit in an Interest and implicit in a Content Object.

Note that a forwarder does not need to know any semantics about a name. It only needs to be able to match a prefix to forward Interests and match an exact or full name to forward Content Objects. It is not sensitive to the name segment types.

The name segment labels specified in this document are given in the table below. Name Segment is a general name segment, typically occurring in the routable prefix and user-specified content name. Other segment types are for functional name components that imply a specific purpose.

Name	Description
Name Segment	A generic name segment that includes arbitrary octets.
Interest Payload ID	An octet string that identifies the payload carried in an Interest. As an example, the Payload ID might be a hash of the Interest Payload. This provides a way to differentiate between Interests based on the Payload solely through a Name Segment without having to include all the extra bytes of the payload itself.
Application Components	An application-specific payload in a name segment. An application may apply its own semantics to these components. A good practice is to identify the application in a Name segment prior to the application component segments.

Table 1: CCNx Name Segment Types

At the lowest level, a Forwarder does not need to understand the semantics of name segments; it need only identify name segment boundaries and be able to compare two name segments (both label and value) for equality. The Forwarder matches paths segment-by-segment against its forwarding table to determine a next hop.

3.1. Name Examples

This section uses the CCNx URI [CCNxURI] representation of CCNx names. Note that as per the message grammar, an Interest must have a Name with at least one name segment and that name segment must have at least 1 octet of value. A Content Object must have a similar name or no name at all. The FIB, on the other hand, could have 0-length names (a default route), or a first name segment with no value, or a regular name.

Name	Description
ccnx:/	A 0-length name, corresponds to a default route.
ccnx:/NAME=	A name with 1 segment of 0 length, distinct from ccnx:/.
ccnx:/NAME=foo/APP:0=bar	A 2-segment name, where the first segment is of type NAME and the second segment is of type APP:0.

Table 2: CCNx Name Examples

3.2. Interest Payload ID

An Interest may also have a Payload which carries state about the Interest but is not used to match a Content Object. If an Interest contains a payload, the Interest name should contain an Interest Payload ID (IPID). The IPID allows a PIT table entry to correctly multiplex Content Objects in response to a specific Interest with a specific payload ID. The IPID could be derived from a hash of the payload or could be a GUID or a nonce. An optional Metadata field defines the IPID field so other systems could verify the IPID, such as when it is derived from a hash of the payload. No system is required to verify the IPID.

4. Cache Control

CCNx supports two fields that affect cache control. These determine how a cache or Content Store handles a Content Object. They are not used in the fast path, but only to determine if a Content Object can be injected on to the fast path in response to an Interest.

The ExpiryTime is a field that exists within the signature envelope of a Validation Algorithm. It is the UTC time in milliseconds after which the Content Object is considered expired and MUST no longer be used to respond to an Interest from a cache. Stale content MAY be flushed from the cache.

The Recommended Cache Time (RCT) is a field that exists outside the signature envelope. It is the UTC time in milliseconds after which the publisher considers the Content Object to be of low value to cache. A cache SHOULD discard it after the RCT, though it MAY keep it and still respond with it. A cache MAY discard the content object

before the RCT time too; there is no contractual obligation to remember anything.

This formulation allows a producer to create a Content Object with a long ExpiryTime but short RCT and keep re-publishing the same, signed, Content Object over and over again by extending the RCT. This allows a form of "phone home" where the publisher wants to periodically see that the content is being used.

5. Content Object Hash

CCNx allows an Interest to restrict a response to a specific hash. The hash covers the Content Object message body and the validation sections, if present. Thus, if a Content Object is signed, its hash includes that signature value. The hash does not include the fixed or hop-by-hop headers of a Content Object. Because it is part of the matching rules (see Section 9), the hash is used at every hop.

There are two options for matching the content object hash restriction in an Interest. First, a forwarder could compute for itself the hash value and compare it to the restriction. This is an expensive operation. The second option is for a border device to compute the hash once and place the value in a header (ConObjHash) that is carried through the network. The second option, of course, removes any security properties from matching the hash, so SHOULD only be used within a trusted domain. The header SHOULD be removed when crossing a trust boundary.

6. Link

A Link is the tuple {Name, [KeyIdRestr], [ContentObjectHashRestr]}. The information in a Link comprises the fields of an Interest which would retrieve the Link target. A Content Object with PayloadType = "Link" is an object whose payload is one or more Links. This tuple may be used as a KeyLink to identify a specific object with the certificate wrapped key. It is RECOMMENDED to include at least one of KeyIdRestr or Content ObjectHashRestr. If neither restriction is present, then any Content Object with a matching name from any publisher could be returned.

7. Hashes

Several protocol fields use cryptographic hash functions, which must be secure against attack and collisions. Because these hash functions change over time, with better ones appearing and old ones falling victim to attacks, it is important that a CCNx protocol implementation supports hash agility.

In this document, we suggest certain hashes (e.g., SHA-256), but a specific implementation may use what it deems best. The normative CCNx Messages [CCNMessages] specification should be taken as the definition of acceptable hash functions and uses.

8. Validation

8.1. Validation Algorithm

The Validator consists of a ValidationAlgorithm that specifies how to verify the message and a ValidationPayload containing the validation output, e.g., the digital signature or MAC. The ValidationAlgorithm section defines the type of algorithm to use and includes any necessary additional information. The validation is calculated from the beginning of the CCNx Message through the end of the ValidationAlgorithm section. The ValidationPayload is the integrity value bytes, such as a MAC or signature.

Some Validators contain a KeyId, identifying the publisher authenticating the Content Object. If an Interest carries a KeyIdRestr, then that KeyIdRestr MUST exactly match the Content Object's KeyId.

Validation Algorithms fall into three categories: MICs, MACs, and Signatures. Validators using Message Integrity Code (MIC) algorithms do not need to provide any additional information; they may be computed and verified based only on the algorithm (e.g., CRC32C). MAC validators require the use of a KeyId identifying the secret key used by the authenticator. Because MACs are usually used between two parties that have already exchanged secret keys via a key exchange protocol, the KeyId may be any agreed-upon value to identify which key is used. Signature validators use public key cryptographic algorithms such as RSA, DSA, ECDSA. The KeyId field in the ValidationAlgorithm identifies the public key used to verify the signature. A signature may optionally include a KeyLocator, as described above, to bundle a Key or Certificate or KeyLink. MAC and Signature validators may also include a SignatureTime, as described above.

A PublicKeyLocator KeyLink points to a Content Object with a DER-encoded X509 certificate in the payload. In this case, the target KeyId must equal the first object's KeyId. The target KeyLocator must include the public key corresponding to the KeyId. That key must validate the target Signature. The payload is an X.509 certificate whose public key must match the target KeyLocator's key. It must be issued by a trusted authority, preferably specifying the valid namespace of the key in the distinguished name.

9. Interest to Content Object matching

A Content Object satisfies an Interest if and only if (a) the Content Object name, if present, exactly matches the Interest name, and (b) the ValidationAlgorithm KeyId of the Content Object exactly equals the Interest KeyIdRestr, if present, and (c) the computed Content ObjectHash exactly equals the Interest ContentObjectHashRestr, if present.

The matching rules are given by this predicate, which if it evaluates true means the Content Object matches the Interest. Ni = Name in Interest (may not be empty), Ki = KeyIdRestr in the interest (may be empty), Hi = ContentObjectHashRestr in Interest (may be empty). Likewise, No, Ko, Ho are those properties in the Content Object, where No and Ko may be empty; Ho always exists (it is an intrinsic property of the Content Object). For binary relations, we use & for AND and | for OR. We use E for the EXISTS (not empty) operator and ! for the NOT EXISTS operator.

As a special case, if the ContentObjectHashRestr in the Interest specifies an unsupported hash algorithm, then no Content Object can match the Interest so the system should drop the Interest and MAY send an InterestReturn to the previous hop. In this case, the predicate below will never get executed because the Interest is never forwarded. If the system is using the optional behavior of having a different system calculate the hash for it, then the system may assume all hash functions are supported and leave it to the other system to accept or reject the Interest.

```
(!No | (Ni=No)) & (!Ki | (Ki=Ko)) & (!Hi | (Hi=Ho)) & (E No | E Hi)
```

As one can see, there are two types of attributes one can match. The first term depends on the existence of the attribute in the Content Object while the next two terms depend on the existence of the attribute in the Interest. The last term is the "Nameless Object" restriction which states that if a Content Object does not have a Name, then it must match the Interest on at least the Hash restriction.

If a Content Object does not carry the Content ObjectHash as an expressed field, it must be calculated in network to match against. It is sufficient within an autonomous system to calculate a Content ObjectHash at a border router and carry it via trusted means within the autonomous system. If a Content Object ValidationAlgorithm does not have a KeyId then the Content Object cannot match an Interest with a KeyIdRestr.

10. Interest Return

This section describes the process whereby a network element may return an Interest message to a previous hop if there is an error processing the Interest. The returned Interest may be further processed at the previous hop or returned towards the Interest origin. When a node returns an Interest it indicates that the previous hop should not expect a response from that node for the Interest -- i.e., there is no PIT entry left at the returning node.

The returned message maintains compatibility with the existing TLV packet format (a fixed header, optional hop-by-hop headers, and the CCNx message body). The returned Interest packet is modified in only two ways:

- The PacketType is set to InterestReturn to indicate a Feedback message.
- The ReturnCode is set to the appropriate value to signal the reason for the return

The specific encodings of the Interest Return are specified in [CCNMessages].

A Forwarder is not required to send any Interest Return messages.

A Forwarder is not required to process any received Interest Return message. If a Forwarder does not process Interest Return messages, it SHOULD silently drop them.

The Interest Return message does not apply to a Content Object or any other message type.

An Interest Return message is a 1-hop message between peers. It is not propagated multiple hops via the FIB. An intermediate node that receives an InterestReturn may take corrective actions or may propagate its own InterestReturn to previous hops as indicated in the reverse path of a PIT entry.

10.1. Message Format

The Interest Return message looks exactly like the original Interest message with the exception of the two modifications mentioned above. The PacketType is set to indicate the message is an InterestReturn and the reserved byte in the Interest header is used as a Return Code. The numeric values for the PacketType and ReturnCodes are in [CCNMessages].

10.2. ReturnCode Types

This section defines the InterestReturn ReturnCode introduced in this RFC. The numeric values used in the packet are defined in [CCNMessages].

Name	Description
No Route (Section 10.3.1)	The returning Forwarder has no route to the Interest name.
HopLimit Exceeded (Section 10.3.2)	The HopLimit has decremented to 0 and need to forward the packet.
Interest MTU too large (Section 10.3.3)	The Interest's MTU does not conform to the required minimum and would require fragmentation.
No Resources (Section 10.3.4)	The node does not have the resources to process the Interest.
Path error (Section 10.3.5)	There was a transmission error when forwarding the Interest along a route (a transient error).
Prohibited (Section 10.3.6)	An administrative setting prohibits processing this Interest.
Congestion (Section 10.3.7)	The Interest was dropped due to congestion (a transient error).
Unsupported Content Object Hash Algorithm (Section 10.3.8)	The Interest was dropped because it requested a Content Object Hash Restriction using a hash algorithm that cannot be computed.
Malformed Interest (Section 10.3.9)	The Interest was dropped because it did not correctly parse.

Table 3: Interest Return Reason Codes

10.3. Interest Return Protocol

This section describes the Forwarder behavior for the various Reason codes for Interest Return. A Forwarder is not required to generate

any of the codes, but if it does, it MUST conform to this specification.

If a Forwarder receives an Interest Return, it SHOULD take these standard corrective actions. A forwarder is allowed to ignore Interest Return messages, in which case its PIT entry would go through normal timeout processes.

- Verify that the Interest Return came from a next-hop to which it actually sent the Interest.
- If a PIT entry for the corresponding Interest does not exist, the Forwarder should ignore the Interest Return.
- If a PIT entry for the corresponding Interest does exist, the Forwarder MAY do one of the following:
  - Try a different forwarding path, if one exists, and discard the Interest Return, or
  - Clear the PIT state and send an Interest Return along the reverse path.

If a forwarder tries alternate routes, it MUST ensure that it does not use same same path multiple times. For example, it could keep track of which next hops it has tried and not re-use them.

If a forwarder tries an alternate route, it may receive a second InterestReturn, possibly of a different type than the first InterestReturn. For example, node A sends an Interest to node B, which sends a No Route return. Node A then tries node C, which sends a Prohibited. Node A should choose what it thinks is the appropriate code to send back to its previous hop

If a forwarder tries an alternate route, it should decrement the Interest Lifetime to account for the time spent thus far processing the Interest.

10.3.1. No Route

If a Forwarder receives an Interest for which it has no route, or for which the only route is back towards the system that sent the Interest, the Forwarder SHOULD generate a "No Route" Interest Return message.

How a forwarder manages the FIB table when it receives a No Route message is implementation dependent. In general, receiving a No Route Interest Return should not cause a forwarder to remove a route.

The dynamic routing protocol that installed the route should correct the route or the administrator who created a static route should correct the configuration. A forwarder could suppress using that next hop for some period of time.

10.3.2. HopLimit Exceeded

A Forwarder MAY choose to send HopLimit Exceeded messages when it receives an Interest that must be forwarded off system and the HopLimit is 0.

10.3.3. Interest MTU Too Large

If a Forwarder receives an Interest whose MTU exceeds the prescribed minimum, it MAY send an "Interest MTU Too Large" message, or it may silently discard the Interest.

If a Forwarder receives an "Interest MTU Too Large" is SHOULD NOT try alternate paths. It SHOULD propagate the Interest Return to its previous hops.

10.3.4. No Resources

If a Forwarder receives an Interest and it cannot process the Interest due to lack of resources, it MAY send an InterestReturn. A lack of resources could be the PIT table is too large, or some other capacity limit.

10.3.5. Path Error

If a forwarder detects an error forwarding an Interest, such as over a reliable link, it MAY send a Path Error Interest Return indicating that it was not able to send or repair a forwarding error.

10.3.6. Prohibited

A forwarder may have administrative policies, such as access control lists, that prohibit receiving or forwarding an Interest. If a forwarder discards an Interest due to a policy, it MAY send a Prohibited InterestReturn to the previous hop. For example, if there is an ACL that says /parc/private can only come from interface e0, but the Forwarder receives one from e1, the Forwarder must have a way to return the Interest with an explanation.

10.3.7. Congestion

If a forwarder discards an Interest due to congestion, it MAY send a Congestion InterestReturn to the previous hop.

10.3.8. Unsupported Content Object Hash Algorithm

If a Content Object Hash Restriction specifies a hash algorithm the forwarder cannot verify, the Interest should not be accepted and the forwarder MAY send an InterestReturn to the previous hop.

10.3.9. Malformed Interest

If a forwarder detects a structural or syntactical error in an Interest, it SHOULD drop the interest and MAY send an InterestReturn to the previous hop. This does not imply that any router must validate the entire structure of an Interest.

11. IANA Considerations

This memo includes no request to IANA.

12. Security Considerations

The CCNx protocol is a layer 3 network protocol, which may also operate as an overlay using other transports, such as UDP or other tunnels. It includes intrinsic support for message authentication via a signature (e.g. RSA or elliptic curve) or message authentication code (e.g. HMAC). In lieu of an authenticator, it may instead use a message integrity check (e.g. SHA or CRC). CCNx does not specify an encryption envelope, that function is left to a high-layer protocol (e.g. [esic]).

The CCNx message format includes the ability to attach MICs (e.g. SHA-256 or CRC), MACs (e.g. HMAC), and Signatures (e.g. RSA or ECDSA) to all packet types. This does not mean that it is a good idea to use an arbitrary ValidationAlgorithm, nor to include computationally expensive algorithms in Interest packets, as that could lead to computational DoS attacks. Applications should use an explicit protocol to guide their use of packet signatures. As a general guideline, an application might use a MIC on an Interest to detect unintentionally corrupted packets. If one wishes to secure an Interest, one should consider using an encrypted wrapper and a protocol that prevents replay attacks, especially if the Interest is being used as an actuator. Simply using an authentication code or signature does not make an Interests secure. There are several examples in the literature on how to secure ICN-style messaging [mobile] [ace].

As a layer 3 protocol, this document does not describe how one arrives at keys or how one trusts keys. The CCNx content object may include a public key embedded in the object or may use the PublicKeyLocator field to point to a public key (or public key certificate) that authenticates the message. One key exchange specification is CCNxKE [ccnxke] [mobile], which is similar to the TLS 1.3 key exchange except it is over the CCNx layer 3 messages. Trust is beyond the scope of a layer-3 protocol protocol and left to applications or application frameworks.

The combination of an ephemeral key exchange (e.g. CCNxKE [ccnxke]) and an encapsulating encryption (e.g. [esic]) provides the equivalent of a TLS tunnel. Intermediate nodes may forward the Interests and Content Objects, but have no visibility inside. It also completely hides the internal names in those used by the encryption layer. This type of tunneling encryption is useful for content that has little or no cache-ability as it can only be used by someone with the ephemeral key. Short term caching may help with lossy links or mobility, but long term caching is usually not of interest.

Broadcast encryption or proxy re-encryption may be useful for content with multiple uses over time or many consumers. There is currently no recommendation for this form of encryption.

The specific encoding of messages will have security implications. [CCNMessages] uses a type-length-value (TLV) encoding. We chose to compromise between extensibility and unambiguous encodings of types and lengths. Some TLVs use variable length T and variable length L fields to accomodate a wide gamut of values while trying to be byte-efficient. Our TLV encoding uses a fixed length 2-byte T and 2-byte L. Using a fixed-length T and L field solves two problems. The first is aliases. If one is able to encode the same value, such as 0x2 and 0x02, in different byte lengths then one must decide if they mean the same thing, if they are different, or if one is illegal. If they are different, then one must always compare on the buffers not the integer equivalents. If one is illegal, then one must validate the TLV encoding -- every field of every packet at every hop. If they are the same, then one has the second problem: how to specify packet filters. For example, if a name has 6 name components, then there are 7 T's and 7 L's, each of which might have up to 4 representations of the same value. That would be 14 fields with 4 encodings each, or 1001 combinations. It also means that one cannot compare, for example, a name via a memory function as one needs to consider that any embedded T or L might have a different format.

The Interest Return message has no authenticator from the previous hop. Therefore, the payload of the Interest Return should only be used locally to match an Interest. A node should never forward that

Interest payload as an Interest. It should also verify that it sent the Interest in the Interest Return to that node and not allow anyone to negate Interest messages.

Caching nodes must take caution when processing content objects. It is essential that the Content Store obey the rules outlined in Section 2.4.3 to avoid certain types of attacks. Unlike NDN, CCNx 1.0 has no mechanism to work around an undesired result from the network (there are no "excludes"), so if a cache becomes poisoned with bad content it might cause problems retrieving content. There are three types of access to content from a content store: unrestricted, signature restricted, and hash restricted. If an Interest has no restrictions, then the requester is not particular about what they get back, so any matching cached object is OK. In the hash restricted case, the requester is very specific about what they want and the content store (and every forward hop) can easily verify that the content matches the request. In the signature verified case (often used for initial manifest discovery), the requester only knows the KeyId that signed the content. It is this case that requires the closest attention in the content store to avoid amplifying bad data. The content store must only respond with a content object if it can verify the signature -- this means either the content object carries the public key inside it or the Interest carries the public key in addition to the KeyId. If that is not the case, then the content store should treat the Interest as a cache miss and let an endpoint respond.

A user-level cache could perform full signature verification by fetching a public key according to the PublicKeyLocator. That is not, however, a burden we wish to impose on the forwarder. A user-level cache could also rely on out-of-band attestation, such as the cache operator only inserting content that it knows has the correct signature.

The CCNx grammar allows for hash algorithm agility via the HashType. It specifies a short list of acceptable hash algorithms that should be implemented at each forwarder. Some hash values only apply to end systems, so updating the hash algorithm does not affect forwarders -- they would simply match the buffer that includes the type-length-hash buffer. Some fields, such as the ConObjHash, must be verified at each hop, so a forwarder (or related system) must know the hash algorithm and it could cause backward compatibility problems if the hash type is updated. [CCNMessages] is the authoritative source for per-field allowed hash types in that encoding.

A CCNx name uses binary matching whereas a URI uses a case insensitive hostname. Some systems may also use case insensitive matching of the URI path to a resource. An implication of this is

that human-entered CCNx names will likely have case or non-ASCII symbol mismatches unless one uses a consistent URI normalization to the CCNx name. It also means that an entity that registers a CCNx routable prefix, say ccnx:/example.com, would need separate registrations for simple variations like ccnx:/Example.com. Unless this is addressed in URI normalization and routing protocol conventions, there could be phishing attacks.

For a more general introduction to ICN-related security concerns and approaches, see [RFC7927] and [RFC7945]

13. References

13.1. 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>.

13.2. Informative References

[ace] Shang, W., Yu, Y., Liang, T., Zhang, B., and L. Zhang, "NDN-ACE: Access control for constrained environments over named data networking", NDN Technical Report NDN-0036, 2015, <http://new.named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf>.

[befrags] Mosko, M. and C. Tschudin, "ICN "Begin-End" Hop by Hop Fragmentation", 2017, <https://www.ietf.org/archive/id/draft-mosko-icnrg-beginendfragment-02.txt>.

[ccnlite] Tschudin, C., et al., University of Basel, "CCN-Lite V2", 2011-2018, <http://www.ccn-lite.net/>.

[CCNMessages] Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format (Internet draft)", 2018, <https://www.ietf.org/id/draft-irtf-icnrg-ccnxmessages-07.txt>.

[ccnxke] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", 2017, <https://www.ietf.org/archive/id/draft-wood-icnrg-ccnxkeyexchange-02.txt>.

[CCNxURI] Mosko, M. and C. Wood, "The CCNx URI Scheme (Internet draft)", 2017, <http://tools.ietf.org/html/draft-mosko-icnrg-ccnxuri-02>.

[chunking]
Mosko, M., "CCNx Content Object Chunking", 2016, <https://www.ietf.org/archive/id/draft-mosko-icnrg-ccnxchunking-02.txt>.

[cicn] Muscariello, L., et al., Cisco Systems, "Community ICN (CICN)", 2017-2018, <https://wiki.fd.io/view/Cicn>.

[dart] Garcia-Luna-Aceves, J. and M. Mirzazad-Barijough, "A Light-Weight Forwarding Plane for Content-Centric Networks", 2016, <https://arxiv.org/pdf/1603.06044.pdf>.

[EpriseNumbers]
IANA, "IANA Private Enterprise Numbers", 2015, <http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>.

[esic] Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", 2017, <https://www.ietf.org/id/draft-wood-icnrg-esic-01.txt>.

[flic] Tschudin, C. and C. Wood, "File-Like ICN Collection (FLIC)", 2017, <https://www.ietf.org/archive/id/draft-tschudin-icnrg-flic-03.txt>.

[mobile] Mosko, M., Uzun, E., and C. Wood, "Mobile Sessions in Content-Centric Networks", IFIP Networking, 2017, <http://dl.ifip.org/db/conf/networking/networking2017/1570334964.pdf>.

[ndn] UCLA, "Named Data Networking", 2007, <http://www.named-data.net>.

[nnc] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", 2009, <http://dx.doi.org/10.1145/1658939.1658941>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC7927] Kutscher, D., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges", 2016, <https://trac.tools.ietf.org/html/rfc7927>.

[RFC7945] Pentikousis, K., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", 2016, <https://trac.tools.ietf.org/html/rfc7945>.

[selectors] Mosko, M., "CCNx Selector Based Discovery", 2017, <https://raw.githubusercontent.com/mmosko/ccnx-protocol-rfc/master/docs/build/draft-mosko-icnrg-selectors-01.txt>.

[terminology] Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology", 2017, <https://www.ietf.org/id/draft-irtf-icnrg-terminology-00.txt>.

[trust] Tschudin, C., Uzun, E., and C. Wood, "Trust in Information-Centric Networking: From Theory to Practice", 2016, <https://doi.org/10.1109/ICCCN.2016.7568589>.

Authors' Addresses

Marc Mosko
PARC, Inc.
Palo Alto, California 94304
USA

Phone: +01 650-812-4405
Email: marc.mosko@parc.com

Ignacio Solis
LinkedIn
Mountain View, California 94043
USA

Email: nsolis@linkedin.com

Christopher A. Wood
University of California Irvine
Irvine, California 92697
USA

Phone: +01 315-806-5939
Email: woodc1@uci.edu

ICN Research Group
Internet-Draft
Intended status: Informational
Expires: January 26, 2021

Prakash Suthar
Milan Stolic
Anil Jangam, Ed.
Cisco Systems Inc.
Dirk Trossen
Huawei Technologies
Ravishankar Ravindran
Sterlite Technologies

July 25, 2020

Native Deployment of ICN in LTE, 4G Mobile Networks
draft-irtf-icnrg-icn-lte-4g-08

Abstract

LTE, 4G mobile networks use IP-based transport for the control plane to establish the data session at the user plane for the actual data delivery. In the existing architecture, IP transport used in the user plane is not optimized for data transport, which leads to inefficient data delivery. For instance, IP unicast routing from server to clients is used for the delivery of multimedia content to User Equipment (UE), with each user receiving a separate stream. From a bandwidth and routing perspective, this approach is inefficient. Multicast and broadcast technologies have recently emerged for mobile networks, but their deployments are very limited or at an experimental stage. ICN is a rapidly emerging technology, although much of the work is focused on fixed networks. The focus of this draft is on native deployment of ICN in cellular mobile networks by using ICN in a 3GPP protocol stack. ICN has inherent capabilities for multicast, anchorless mobility, and security, while being optimized for data delivery using local caching at the edge. The proposed approaches in this draft allow ICN to be enabled natively over the current LTE stack comprising PDCP/RLC/MAC/PHY, or in a dual stack mode (alongside IP). 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/.

Table of Contents

1. Introduction

LTE mobile technology is built as an all-IP network. It uses IP routing protocols (OSPF, ISIS, BGP, etc.) to establish network routes over which route data traffic. Stickiness of an IP address to a device is the key to get connected to a mobile network. The same IP address is maintained through the session until the device gets detached or moves to another network.

Key protocols used in 4G and LTE networks are GPRS Tunneling protocol (GTP), DIAMETER, and other protocols that are built on top of IP. One of the biggest challenges with IP-based routing in LTE is that it is not optimized for data transport. As an alternative to IP routing, this draft presents instead the native implementation of Information Centric Networking (ICN) in 3GPP, offering an opportunity to leverage inherent ICN capabilities such as in-network caching, multicast, anchorless mobility management, and authentication. This draft proposes options for deploying ICN in mobile networks, and how those options affect mobile providers and end users.

This document represents the consensus of the Information-Centric Networking Research Group (ICNRG). It has been reviewed extensively by the Research Group (RG) members active in the specific areas of work covered by the document.

2. 3GPP Terminology and Concepts

1. Access Point Name

The Access Point Name (APN) is a Fully Qualified Domain Name (FQDN) and resolves to a set of gateways in an operator's network. APN identifies the packet data network (PDN) with which a mobile data user wants to communicate. In addition to identifying a PDN, an APN may also be used to define the type of service, QoS, and other logical entities inside GGSN, PGW.

2. Control Plane

The control plane carries signaling traffic and is responsible for routing between eNodeB and MME, MME and HSS, MME and SGW, SGW and PGW, etc. Control plane signaling is required to authenticate and authorize UE and establish a mobility session with mobile gateways (SGW/PGW). Control plane functions also include system configuration and management.

3. Dual Address PDN/PDP Type

   The dual address Packet Data Network/Packet Data Protocol (PDN/PDP) Type (IPv4v6) is used in 3GPP context, in many cases as a synonym for dual stack; i.e., a connection type capable of serving IPv4 and IPv6 simultaneously.

4. eNodeB

   The eNodeB is a base station entity that supports the Long-Term Evolution (LTE) air interface.

5. Evolved Packet Core

   The Evolved Packet Core (EPC) is an evolution of the 3GPP GPRS system characterized by a higher-data-rate, lower-latency, packet-optimized system. The EPC comprises some sub components of the EPS core such as Mobility Management Entity (MME), Serving Gateway (SGW), Packet Data Network Gateway (PDN-GW), and Home Subscriber Server (HSS).

6. Evolved Packet System

   The Evolved Packet System (EPS) is an evolution of the 3GPP GPRS system characterized by a higher-data-rate, lower-latency, packet-optimized system that supports multiple Radio Access Technologies (RATs). The EPS comprises the EPC together with the Evolved Universal Terrestrial Radio Access (E-UTRA) and the Evolved Universal Terrestrial Radio Access Network (E-UTRAN).

7. Evolved UTRAN

   The E-UTRAN is a communications network sometimes referred to as 4G, and consists of eNodeB (4G base stations). The E-UTRAN allows connectivity between the User Equipment and the core network.

8. GPRS Tunneling Protocol

   The GPRS Tunneling Protocol (GTP) [TS29.060] [TS29.274] [TS29.281] is a tunneling protocol defined by 3GPP. It is a network-based mobility protocol, working similar to Proxy Mobile IPv6 (PMIPv6). However, GTP also provides functionality beyond mobility, such as in-band signaling related to QoS and charging, among others.

9. Gateway GPRS Support Node

The Gateway GPRS Support Node (GGSN) is a gateway function in the GPRS and 3G network that provides connectivity to the Internet or other PDNs. The host attaches to a GGSN identified by an APN assigned to it by an operator. The GGSN also serves as the topological anchor for addresses/prefixes assigned to the User Equipment.

10. General Packet Radio Service

The General Packet Radio Service (GPRS) is a packet-oriented mobile data service available to users of the 2G and 3G cellular communication systems--the GSM--specified by 3GPP.

11. Home Subscriber Server

The Home Subscriber Server (HSS) is a database for a given subscriber and was introduced in 3GPP Release-5. It is the entity containing subscription-related information to support the network entities that handle calls/sessions.

12. Mobility Management Entity

The Mobility Management Entity (MME) is a network element responsible for control plane functionalities, including authentication, authorization, bearer management, layer-2 mobility, and so on. The MME is essentially the control plane part of the SGSN in the GPRS. The user plane traffic bypasses the MME.

13. Public Land Mobile Network

The Public Land Mobile Network (PLMN) is a network operated by a single administration. A PLMN (and, therefore, also an operator) is identified by the Mobile Country Code (MCC) and the Mobile Network Code (MNC). Each (telecommunications) operator providing mobile services has its own PLMN.

14. Policy and Charging Control

The Policy and Charging Control (PCC) framework is used for QoS policy and charging control. It has two main functions: flow-based charging (including online credit control), and policy control (for example, gating control, QoS control, and QoS signaling). It is optional to 3GPP EPS but needed if dynamic policy and charging control by means of PCC rules based on user and services are desired.

15. Packet Data Network

The Packet Data Network (PDN) is a packet-based network that either belongs to the operator or is an external network such as the Internet or a corporate intranet. The user eventually accesses services in one or more PDNs. The operator's packet core networks are separated from packet data networks either by GGSNs or PDN Gateways (PGWs).

16. Serving Gateway

The Serving Gateway (SGW) is a gateway function in the EPS, which terminates the interface towards the E-UTRAN. The SGW is the Mobility Anchor point for layer-2 mobility (inter-eNodeB handovers). For each UE connected with the EPS, there is only one SGW at any given point in time. The SGW is essentially the user plane part of the GPRS's SGSN.

17. Packet Data Network Gateway

The Packet Data Network Gateway (PGW) is a gateway function in the Evolved Packet System (EPS), which provides connectivity to the Internet or other PDNs. The host attaches to a PGW identified by an APN assigned to it by an operator. The PGW also serves as the topological anchor for addresses/prefixes assigned to the User Equipment.

18. Packet Data Protocol Context

A Packet Data Protocol (PDP) context is the equivalent of a virtual connection between the User Equipment (UE) and a PDN using a specific gateway.

19. Packet Data Protocol Type

A Packet Data Protocol Type (PDP Type) identifies the used/ allowed protocols within the PDP context. Examples are IPv4, IPv6, and IPv4v6 (dual-stack).

20. Serving GPRS Support Node

The Serving GPRS Support Node (SGSN) is a network element located between the radio access network (RAN) and the gateway (GGSN). A per-UE point-to-point (p2p) tunnel between the GGSN and SGSN transports the packets between the UE and the gateway.

21. Terminal Equipment

The Terminal Equipment (TE) is any device/host connected to the Mobile Terminal (MT) offering services to the user. A TE may

communicate to an MT, for example, over the Point-to-Point Protocol (PPP).

22. UE, MS, MN, and Mobile

The terms User Equipment (UE), Mobile Station (MS), Mobile Node (MN), and mobile refer to the devices that are hosts with the ability to obtain Internet connectivity via a 3GPP network. An MS comprises the Terminal Equipment (TE) and a Mobile Terminal (MT). The terms UE, MS, MN, and mobile are used interchangeably within this document.

23. User Plane

The user plane refers to data traffic and the required bearers for the data traffic. In practice, IP is the only data traffic protocol used in the user plane.

3. LTE, 4G Mobile Network

3.1. Network Overview

With the introduction of LTE, mobile networks moved to all-IP transport for all elements such as eNodeB, MME, SGW/PGW, HSS, PCRF, routing and switching, etc. Although the LTE network is data-centric, it has support for legacy Circuit Switch features such as voice and SMS through transitional CS fallback and flexible IMS deployment [GRAYSON]. For each mobile device attached to the radio (eNodeB), there is a separate overlay tunnel (GPRS Tunneling Protocol, GTP) between eNodeB and Mobile gateways (i.e., SGW, PGW).

When any UE is powered up, it attaches to a mobile network based on its configuration and subscription. After a successful attachment procedure, the UE registers with the mobile core network, and an IPv4 and/or IPv6 address is assigned. A default bearer is created for each UE, and it is assigned to default Access Point Name (APN).

The GTP tunnel is used to carry user traffic between gateways and mobile devices, therefore mandating unicast delivery for any data transfer. It is also important to understand the overhead of GTP and IPSec protocols. All mobile backhaul traffic is encapsulated using a GTP tunnel, which has overhead of 8 bytes on top of IP and UDP [NGMN]. Additionally, if IPSec is used for security (which is often required if the Service Provider is using a shared backhaul), it adds overhead based on the IPSec tunneling model (tunnel or transport) as well as the encryption and authentication header algorithm used. If we consider as an example an Advanced Encryption Standard (AES)

encryption, the overhead can be significant [OLTEANU], particularly for smaller payloads.

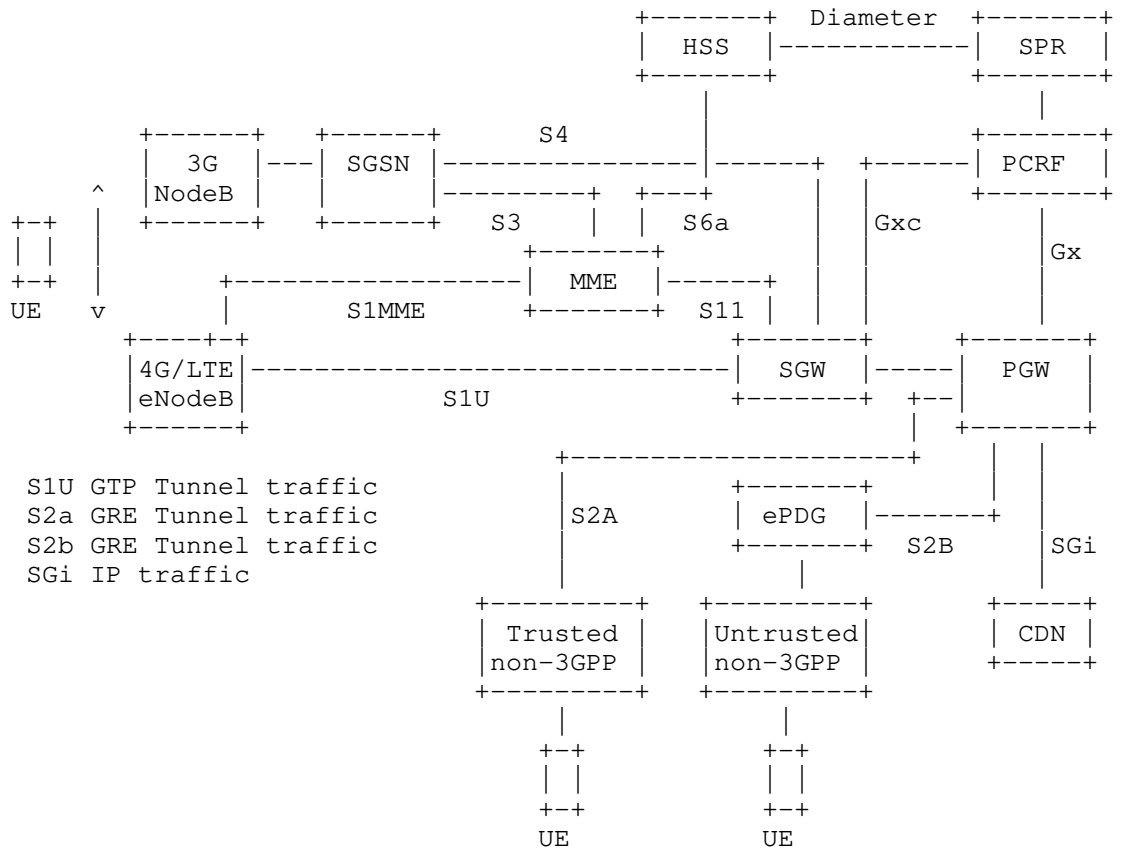


Figure 1: LTE, 4G Mobile Network Overview

If we consider the combined impact of GTP, IPSec and unicast traffic, the data delivery is not efficient. The IETF has developed various header compression algorithms to reduce the overhead associated with IP packets. Some techniques are robust header compression (ROHC) and enhanced compression of the real-time transport protocol (ECRTP) so that the impact of overhead created by GTP, IPsec, etc., is reduced to some extent [BROWER]. For commercial mobile networks, 3GPP has adopted different mechanisms for header compression to achieve efficiency in data delivery [TS25.323]; those solutions can be adapted to other data protocols, such as ICN, too [ICNLOWPAN] [TLVCOMP].

3.2. QoS Challenges

During the attachment procedure, a default bearer is created for each UE and it is assigned to the default Access Point Name (APN). The QoS values assigned during the initial attach are best-effort, with no guarantees. Additional dedicated bearer(s) with enhanced QoS parameters are established, depending on specific application needs.

While all traffic within a certain bearer receives the same treatment, QoS parameters supporting these requirements can be very granular in different bearers. These values vary for the control, management and user traffic, and can be very different depending on application key parameters such as latency, jitter (important for voice and other real-time applications), packet loss, and queuing mechanism (strict priority, low-latency, fair, and so on).

Implementation of QoS for mobile networks is done at two stages: at content prioritization/marking and transport marking, and congestion management. From the transport perspective, QoS is defined at layer 2 as class of service (CoS) and at layer 3 either as DiffServ code point (DSCP) or type of service (ToS). The mapping of DSCP to CoS takes place at layer 2/3 switching and routing elements. 3GPP has a specified a QoS Class Identifier (QCI), which represents different types of content and equivalent mappings to the DSCP at transport layer [TS23.401]. However, this requires manual configuration at different elements and is therefore prone to possible misconfigurations.

In summary, QoS configuration in mobile networks for user plane traffic requires synchronization of parameters among different platforms. Normally, QoS in IP is implemented using DiffServ, which uses hop-by-hop QoS configuration at each router. Any inconsistency in IP QoS configuration at routers in the forwarding path can result in a poor subscriber experience (e.g., packet classified as high priority can go to a lower priority queue). By deploying ICN, we intend to enhance the subscriber experience using policy-based configuration, which can be associated with the named contents [ICNQoS] at the ICN forwarder. Further investigation is needed to understand how QoS in ICN can be implemented to meet the IP QoS requirements [RFC4594].

Research papers published so far explore the possibility of classifications based on name prefixes (thus addressing the problem of IP QoS not being information aware), or on popularity or placement (looking at a distance of a content from a requester). However, focus of these research efforts is on faster routing of Interest requests towards the content rather than content delivery.

3.3. Data Transport Using IP

The data delivered to mobile devices is sent in unicast semantic inside the GTP tunnel from an eNodeB to a PDN gateway (PGW), as described in 3GPP specifications [TS23.401]. While the technology exists to address the issue of possible multicast delivery, there are many difficulties related to multicast protocol implementations on the RAN side of the network. Transport networks in the backhaul and core addressed the multicast delivery a long time ago and have implemented it in most cases in their multi-purpose integrated transport. But the RAN part of the network is still lagging behind due to complexities related to client mobility, handovers, and the fact that the potential gain to Service Providers may not justify the investment, which explains the prevalence of unicast delivery in mobile networks. Techniques to handle multicast (such as LTE-B or eMBMS) have been designed to handle pre-planned content delivery, such as live content, which contrasts user behavior today, largely based on content (or video) on demand model.

To ease the burden on the bandwidth of the SGi interface, caching is introduced in a similar manner as with many Enterprises. In mobile networks, whenever possible, cached data is delivered. Caching servers are placed at a centralized location, typically in the Service Provider's Data Center, or in some cases lightly distributed in Packet Core locations with the PGW nodes close to the Internet and IP services access (SGi interface). This is a very inefficient concept because traffic must traverse the entire backhaul path for the data to be delivered to the end user. Other issues, such as out-of-order delivery, contribute to this complexity and inefficiency, which needs to be addressed at the application level.

3.4. Virtualizing Mobile Networks

The Mobile packet core deployed in a major Service Provider network is either based on dedicated hardware or, in some cases, large capacity x86 platforms. With the adoption of Mobile Virtual Network Operators (MVNO), public safety networks, and enterprise mobility networks, elastic mobile core architecture are needed. By deploying the mobile packet core on a commercially off-the-shelf (COTS) platform, using a virtualized infrastructure (NFVI) framework and end-to-end orchestration, new deployments can be simplified to provide optimized TCO.

While virtualization is growing, and many mobile providers use a hybrid architecture that consists of dedicated and virtualized infrastructures, the control and data planes are still the same. There is also work under way to separate the control and user plane for the network to scale better. Virtualized mobile networks and

network slicing with control and user plane separation provide a mechanism to evolve the GTP-based architecture towards an Openflow SDN-based signaling for LTE and proposed 5G core. Some early architecture work for 5G mobile technologies provides a mechanism for control and user plane separation and simplifies the mobility call flow by introducing Openflow-based signaling [ICN5G]. This has been considered by 3GPP [EPCCUPS] and is also described in [SDN5G].

4. Data Transport Using ICN

For mobile devices, the edge connectivity is between mobile terminal and a router or mobile edge computing (MEC) [MECSPEC] element. Edge computing has the capability of processing client requests and segregating control and user traffic at the edge of radio, rather than sending all requests to the mobile gateway.

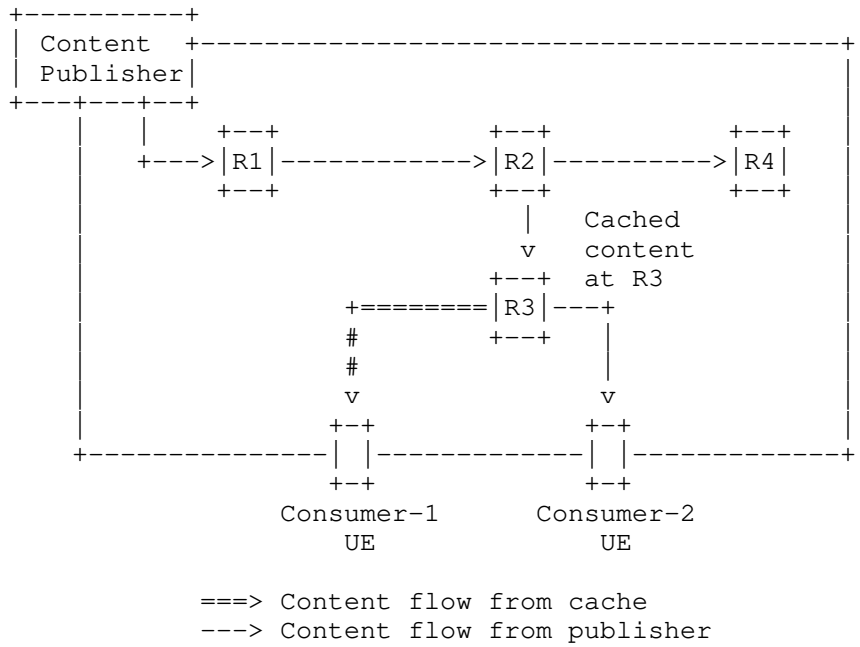


Figure 2: ICN Architecture

Edge computing transforms radio access network into an intelligent service edge capable of delivering services directly from the edge of the network, while providing the best possible performance to the client. Edge computing can be an ideal candidate for implementing ICN forwarders in addition to its usual function of managing mobile

termination. In addition to edge computing, other transport elements, such as routers, can work as ICN forwarders.

Data transport using ICN is different to IP-based transport by introducing uniquely named-data as a core design principle. Communication in ICN takes place between the content provider (producer) and the end user (consumer), as described in Figure 2.

Every node in a physical path between a client and a content provider is called the ICN forwarder or router. It can route the request intelligently and cache content so it can be delivered locally for subsequent requests from any other client. For mobile networks, transport between a client and a content provider consists of radio network + mobile backhaul and IP core transport + Mobile Gateways + Internet + content data network (CDN).

To understand the suitability of ICN for mobile networks, we will discuss the ICN framework by describing its protocols architecture and different types of messages to then consider how we can use this in mobile networks for delivering content more efficiently. ICN uses two types of packets called "interest packet" and "data packet" as described in Figure 3.

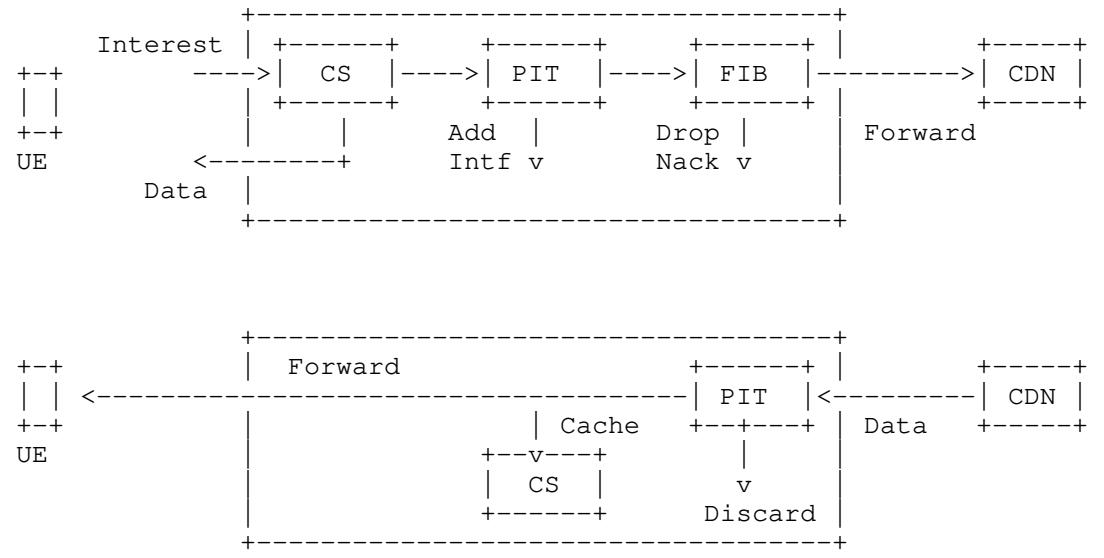


Figure 3: ICN Interest, Data Packet and Forwarder

In an LTE network, when a mobile device wants to receive certain content, it will send an Interest message to the closest eNodeB.

Interest packets follow the TLV format [RFC8609] and contain mandatory fields, such as name of the content and content matching restrictions (KeyIdRestr and ContentObjectHashRestr), expressed as a tuple [RFC8569]. The content matching tuple uniquely identifies the matching data packet for the given Interest packet. Another attribute called HopLimit is used to detect looping Interest messages.

An ICN router will receive an Interest packet and lookup if a request for such content has arrived earlier from another client. If so, it may be served from the local cache; otherwise, the request is forwarded to the next-hop ICN router. Each ICN router maintains three data structures: Pending Interest Table (PIT), Forwarding Information Base (FIB), and Content Store (CS). The Interest packet travels hop-by-hop towards the content provider. Once the Interest packet reaches the content provider; it will return a Data packet containing information such as content name, signature, and the actual data.

The data packet travels in reverse direction following the same path taken by the Interest packet, maintaining routing symmetry. Details about algorithms used in PIT, FIB, CS, and security trust models are described in various resources [CCN]; here, we have explained the concept and its applicability to the LTE network.

5. ICN Deployment in 4G and LTE Networks

5.1. General ICN Deployment Considerations

In LTE/4G mobile networks, both user and control plane traffic have to be transported from the edge to the mobile packet core via IP transport. The evolution of the existing mobile packet core using Control and User Plane Separation (CUPS) [TS23.714] enables flexible network deployment and operation by distributed deployment and the independent scaling of control plane and user plane functions - while not affecting the functionality of existing nodes subject to this split.

In the CUPS architecture, there is an opportunity to shorten the path for user plane traffic by deploying offload nodes closer to the edge [OFFLOAD]. With this major architecture change, a User Plane Function (UPF) node is placed close to the edge so traffic no longer needs to traverse the entire backhaul path to reach the EPC. In many cases, where feasible, the UPF can be collocated with the eNodeB, which is also a business decision based on user demand. Placing a Publisher close to the offload site, or at the offload site, provides for a significant improvement in user experience, especially with latency-sensitive applications. This capability allows for the

introduction of ICN and amplifies its advantages. This section analyzes the potential impact of ICN on control and user plane traffic for centralized and disaggregate CUPS-based mobile network architecture.

5.2. ICN Deployment Scenarios

The deployment of ICN provides an opportunity to further optimize the existing data transport in LTE/4G mobile networks. The various deployment options that ICN and IP provide are somewhat analogous to the deployment scenarios when IPv6 was introduced to interoperate with IPv4 except, with ICN, the whole IP stack is being replaced. We have reviewed [RFC6459] and analyzed the impact of ICN on control plane signaling and user plane data delivery. In general, ICN can be deployed by natively replacing IP transport (IPv4 and IPv6) or as an overlay protocol. Figure 4 describes a modified protocol stack to support ICN deployment scenarios.

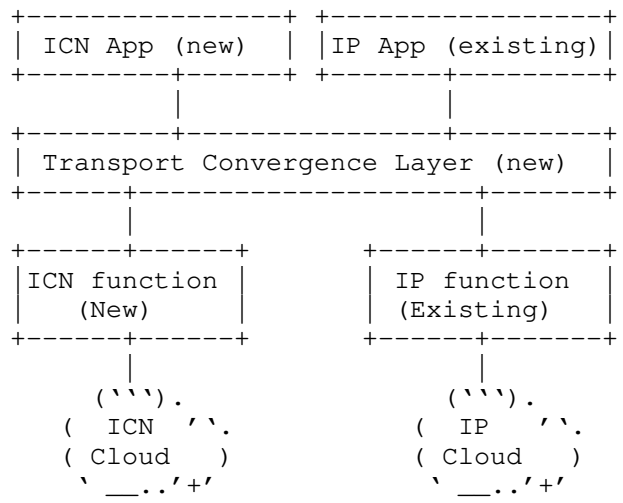


Figure 4: IP/ICN Convergence and Deployment Scenarios

As shown in Figure 4, for applications running either in the UE or in the content provider system to use the new transport option, we propose a new transport convergence layer (TCL). This transport convergence layer helps determine the type of transport (such as ICN or IP), as well as the type of radio interface (LTE or WiFi or both) used to send and receive traffic based on preference (e.g., content location, content type, content publisher, congestion, cost, QoS). It helps configure and determine the type of connection as well as

the overlay mode (ICNoIP or IPoICN) between application and the protocol stack (IP or ICN) to be used.

Combined with the existing IP function, the ICN function provides support for either native ICN and/or the dual stack (ICN/IP) transport functionality. See Section 5.4.1 for elaborate descriptions of these functional layers.

The TCL can use several mechanisms for transport selection. It can use a per-application configuration through a management interface, possibly even a user-facing setting realized through a user interface, like those used today that select cellular over WiFi being used for selected applications. In another option, it might use a software API, which an adapted IP application could use to specify (such as an ICN transport) for obtaining its benefits.

Another potential application of TCL is in implementation of network slicing, with a slice management capability locally or through an interface to an external slice manager via an API [GALIS]. This solution can enable network slicing for IP and ICN transport selection from the UE itself. The TCL could apply slice settings to direct certain traffic (or applications) over one slice and others over another slice, determined by some form of 'slicing policy'. Slicing policy can be obtained externally from the slice manager or configured locally on UE.

From the perspective of applications either on the UE or at a content provider, the following options are possible for ICN deployment natively and/or with IP.

1. IP over IP

   In this scenario, the UE uses applications tightly integrated with the existing IP transport infrastructure. The TCL has no additional function because packets are forwarded directly using an IP protocol stack, which sends packets over the IP transport.

2. ICN over ICN

   Similar to case 1, ICN applications integrate tightly with the ICN transport infrastructure. The TCL has no additional responsibility because packets are forwarded directly using ICN protocol stack, which sends packets over the ICN transport.

3. ICN over IP (ICNoIP)

   In this scenario, the underlying IP transport infrastructure is not impacted (that is, ICN is implemented as an IP overlay

between user equipment (UE) and content provider). IP routing is used from Radio Access Network (eNodeB) to the mobile backhaul, the IP core, and the Mobile Gateway (SGW/PGW). The UE attaches to the Mobile Gateway (SGW/PGW) using an IP address. Also, the data transport between Mobile Gateway (SGW/PGW) and content publisher uses IP. The content provider can serve content either using IP or ICN, based on the UE request.

An approach to implement ICN in mobile backhaul networks is described in [MBICN]. It implements a GTP-U extension header option to encapsulate ICN payload in a GTP tunnel. However, as this design runs ICN as an IP overlay, the mobile backhaul can be deployed using native IP. The proposal describes a mechanism where the GTP-U tunnel can be terminated by hair pinning the packet before it reaches SGW, if an ICN-enabled node is deployed in the mobile backhaul (that is, between eNodeB and SGW). This could be useful when an ICN data packet is stored in the ICN node (such as repositories, caches) in the tunnel path so that the ICN node can reply without going all the way through the mobile core. While a GTP-U extension header is used to carry UE specific ICN payload, they are not visible to the transport, including SGW. On the other hand, the PGW can use the UE-specific ICN header extension and ICN payload to set up an uplink transport towards a content provider in the Internet. In addition, the design assumes a proxy function at the edge, to perform ICN data retrieval on behalf of a non-ICN end device.

4. IP over ICN (IPoICN)

[IPoICN] provides an architectural framework for deployment of IP as an overlay over ICN protocol. Implementing IP services over ICN provides an opportunity to leverage the benefits of ICN in the transport infrastructure while there is no impact on end devices (UE and access network) as they continue to use IP. IPoICN however, will require an inter-working function (IWF/ Border Gateway) to translate various transport primitives. The IWF function will provide a mechanism for protocol translation between IPoICN and native IP deployment for mobile network. After reviewing [IPoICN], we understand and interpret that ICN is implemented in the transport natively; however, IP is implemented in UE, eNodeB, and Mobile gateway (SGW/PGW), which is also called as a network attach point (NAP).

For this, said NAP receives an incoming IP or HTTP packet (the latter through TCP connection termination) and publishes the packet under a suitable ICN name (i.e., the hash over the destination IP address for an IP packet or the hash over the FQDN of the HTTP request for an HTTP packet) to the ICN network. In

the HTTP case, the NAP maintains a pending request mapping table to map returning responses to the terminated TCP connection.

5. Hybrid ICN (hICN)

An alternative approach to implement ICN over IP is provided in Hybrid ICN [HICN]. It describes a novel approach to integrate ICN into IPv6 without creating overlays with a new packet format as an encapsulation. hICN addresses the content by encoding a location-independent name in an IPv6 address. It uses two name components--name prefix and name suffix--that identify the source of data and the data segment within the scope of the name prefix, respectively.

At application layer, hICN maps the name into an IPv6 prefix and, thus, uses IP as transport. As long as the name prefixes, which are routable IP prefixes, point towards a mobile GW (PGW or local breakout, such as CUPS), there are potentially no updates required to any of the mobile core gateways (for example, SGW/ PGW). The IPv6 backhaul routes the packets within the mobile core. hICN can run in the UE, in the eNodeB, in the mobile backhaul, or in the mobile core. Finally, as hICN itself uses IPv6, it cannot be considered as an alternative transport layer.

5.3. ICN Deployment in LTE Control Plane

In this section, we analyze signaling messages that are required for different procedures, such as attach, handover, tracking area update, and so on. The goal of this analysis is to see if there are any benefits to replacing IP-based protocols with ICN for LTE signaling in the current architecture. It is important to understand the concept of point of attachment (POA). When UE connects to a network, it has the following POAs:

1. eNodeB managing location or physical POA

2. Authentication and Authorization (MME, HSS) managing identity or authentication POA

3. Mobile Gateways (SGW, PGW) managing logical or session management POA

In the current architecture, IP transport is used for all messages associated with the control plane for mobility and session management. IP is embedded very deeply into these messages utilizing TLV syntax for carrying additional attributes such as a layer 3 transport. The physical POA in the eNodeB handles both mobility and session management for any UE attached to 4G, LTE network. The

number of mobility management messages between different nodes in an LTE network per signaling procedure is shown in Table 1.

Normally, two types of UE devices attach to the LTE network: SIM based (need 3GPP mobility protocol for authentication) or non-SIM based (which connect to WiFi network). Both device types require authentication. For non-SIM based devices, AAA is used for authentication. We do not propose to change UE authentication or mobility management messaging for user data transport using ICN. A separate study would be required to analyze the impact of ICN on mobility management messages structures and flows. We are merely analyzing the viability of implementing ICN as a transport for control plane messages.

It is important to note that, if MME and HSS do not support ICN transport, they still need to support UE capable of dual stack or native ICN. When UE initiates an attach request using the identity as ICN, MME must be able to parse that message and create a session. MME forwards UE authentication to HSS, so HSS must be able to authenticate an ICN-capable UE and authorize create session [TS23.401].

LTE Signaling Procedures	MME	HSS	SGW	PGW	PCRF
Attach	10	2	3	2	1
Additional default bearer	4	0	3	2	1
Dedicated bearer	2	0	2	2	0
Idle-to-connect	3	0	1	0	0
Connect-to-Idle	3	0	1	0	0
X2 handover	2	0	1	0	0
S1 handover	8	0	3	0	0
Tracking area update	2	2	0	0	0
Total	34	2	14	6	3

Table 1: Signaling Messages in LTE Gateways

Anchorless mobility [ALM] provides a fully decentralized, control-plane agnostic solution to handle producer mobility in ICN. Mobility management at layer-3 level makes it access agnostic and transparent to the end device or the application. The solution discusses handling mobility without having to depend on core network functions (e.g. MME); however, a location update to the core network may still be required to support legal compliance requirements such as lawful intercept and emergency services. These aspects are open for further study.

One of the advantages of ICN is in the caching and reusing of the content, which does not apply to the transactional signaling exchange. After analyzing LTE signaling call flows [TS23.401] and messages inter-dependencies (see Table 1), our recommendation is that it is not beneficial to deploy ICN for control plane and mobility management functions. Among the features of ICN design, Interest aggregation and content caching are not applicable to control plane signaling messages. Control plane messages are originated and consumed by the applications and they cannot be shared.

5.4. ICN Deployment in LTE User Plane

We will consider Figure 1 to discuss different mechanisms to deploy ICN in mobile networks. In Section 5.2, we discussed generic deployment scenarios of ICN. In this section, we discuss the specific use cases of native ICN deployment in LTE user plane. We consider the following options:

1. Dual stack ICN deployment in UE

2. Native ICN deployments in UE

3. ICN deployment in eNodeB

4. ICN deployment in mobile gateways (SGW/PGW)

5.4.1. Dual stack ICN deployments in UE

The control and user plane communications in LTE, 4G mobile networks are specified in 3GPP documents [TS23.203] and [TS23.401]. It is important to understand that UE can be either consumer (receiving content) or publisher (pushing content for other clients). The protocol stack inside the mobile device (UE) is complex because it must support multiple radio connectivity access to eNodeB(s).

Figure 5 provides a high-level description of a protocol stack, where IP is defined at two layers: (1) user plane communication and (2) UDP encapsulation. User plane communication takes place between Packet Data Convergence Protocol (PDCP) and Application layer, whereas UDP encapsulation is at GTP protocol stack.

The protocol interactions and impact of supporting tunneling of ICN packet into IP or to support ICN natively are described in Figure 5 and Figure 6, respectively.

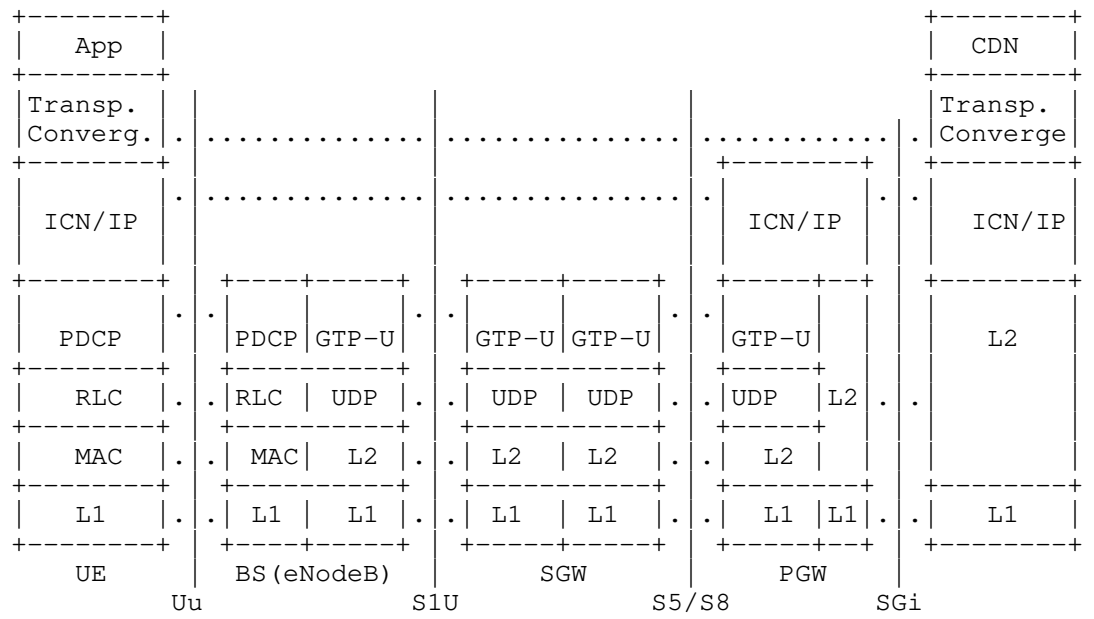


Figure 5: Dual Stack ICN Deployment in UE

The protocols and software stack used inside LTE capable UE support both 3G and LTE software interworking and handover. the latest 3GPP Rel.13 onward specification describes the use of IP and non-IP protocols to establish logical/session connectivity. We intend to leverage the non-IP protocol-based mechanism to deploy ICN protocol stack in UE, as well as in eNodeB and mobile gateways (SGW, PGW).

1. An existing application layer can be modified to provide options for a new ICN-based application and existing IP-based applications. The UE can continue to support existing IP-based applications or host new applications developed to support native ICN as transport, ICNoIP, or IPoICN-based transport. The application layer has the option of selecting either ICN or IP transport, as well as radio interface, to send and receive data traffic.

   Our proposal is to provide an Application Programming Interface (API) to the application developers so they can choose either ICN or IP transport for exchanging the traffic with the network. As mentioned in Section 5.2, the transport convergence layer (TCL) function handles the interaction of applications with multiple transport options.

2. The transport convergence layer helps determine the type of transport (such as ICN, hICN, or IP) and type of radio interface (LTE or WiFi, or both) used to send and receive traffic. Application layer can make the decision to select a specific transport based on preference, such as content location, content type, content publisher, congestion, cost, QoS, and so on. There can be an Application Programming Interface (API) to exchange parameters required for transport selection. Southbound interactions of Transport Convergence Layer (TCL) will be either to IP or ICN at the network layer.

   When selecting the IPoICN mode, the TCL is responsible for receiving an incoming IP or HTTP packet and publishing the packet to the ICN network under a suitable ICN name (that is, the hash over the destination IP address for an IP packet, or the hash over the FQDN of the HTTP request for an HTTP packet). In the HTTP case, the TCL maintains a pending request mapping table to map returning responses to the originating HTTP request. The common API will provide a 'connection' abstraction for this HTTP mode of operation, returning the response over said connection abstraction, akin to the TCP socket interface, while implementing a reliable transport connection semantic over the ICN from the UE to the receiving UE or the PGW. If the HTTP protocol stack remains unchanged, therefore utilizing the TCP protocol for transfer, the TCL operates in local TCP termination mode, retrieving the HTTP packet through said local termination.

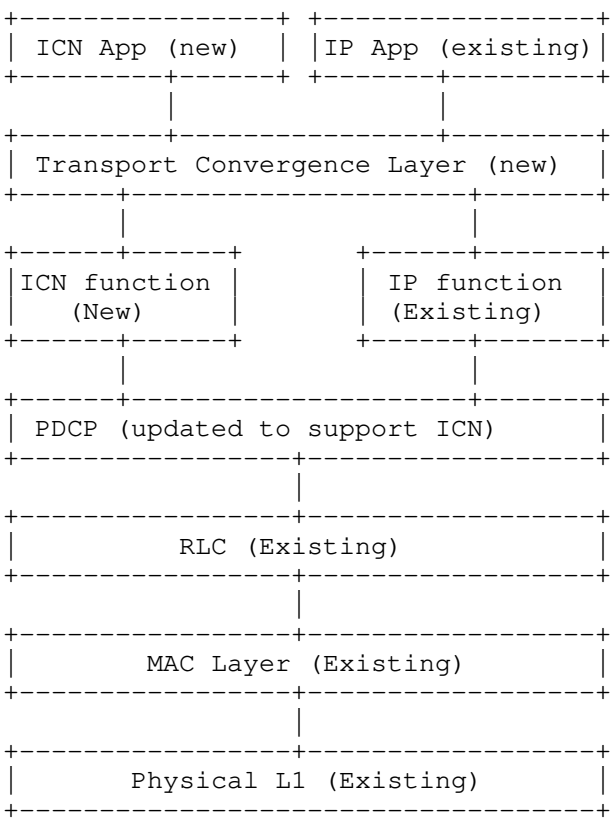


Figure 6: Dual Stack ICN Protocol Interactions

3. The ICN function (forwarder) is introduced in parallel to the existing IP layer. The ICN forwarder contains functional capabilities to forward ICN packets, such as an Interest packet to eNodeB or a response "data packet" from eNodeB to the application.

4. For the dual-stack scenario, when UE is not supporting ICN as transport, we use an IP underlay to transport ICN packets. The ICN function will use the IP interface to send Interest and Data packets for fetching or sending data using ICN protocol function. This interface will use the ICN overlay over IP using any overlay tunneling mechanism.

5. To support ICN at network layer in UE, the PDCP layer must be aware of ICN capabilities and parameters. PDCP is located in the Radio Protocol Stack in the LTE Air interface, between IP

(Network layer) and Radio Link Control Layer (RLC). PDCP performs the following functions [TS36.323]:

1. Data transport by listening to upper layer, formatting and pushing down to Radio Link Layer (RLC)

2. Header compression and decompression using Robust Header Compression (ROHC)

3. Security protections such as ciphering, deciphering, and integrity protection

4. Radio layer messages associated with sequencing, packet drop detection and re-transmission, and so on.

6. No changes are required for lower layer such as RLC, MAC, and Physical (L1) because they are not IP aware.

One key point to understand in this scenario is that ICN is deployed as an overlay on top of IP.

5.4.2. Native ICN Deployments in UE

We propose to implement ICN natively in UE by modifying the PDCP layer in 3GPP protocols. Figure 7 provides a high-level protocol stack description where ICN is used at the following different layers:

1. User plane communication

2. Transport layer

User plane communication takes place between PDCP and application layer, whereas ICN transport is a substitute of the GTP protocol. The removal of the GTP protocol stack is a significant change in the mobile architecture because GTP is used not just for routing but for mobility management functions, such as billing, mediation, and policy enforcement.

If we implement ICN natively in the UE, the communication between UE and eNodeB will change. Also, this will avoid tunneling the user plane traffic from eNodeB to the mobile packet core (SGW, PGW) using a GTP tunnel.

For native ICN deployment, an application will be configured to use ICN forwarder so there is no need for Transport Convergence. Also, to support ICN at the network layer in UE, we need to modify the existing PDCP layer to be aware of ICN capabilities and parameters.

The native implementation will also provide opportunities to develop new use cases leveraging ICN capabilities, such as seamless mobility, UE to UE content delivery using radio network without traversing the mobile gateways, and more.

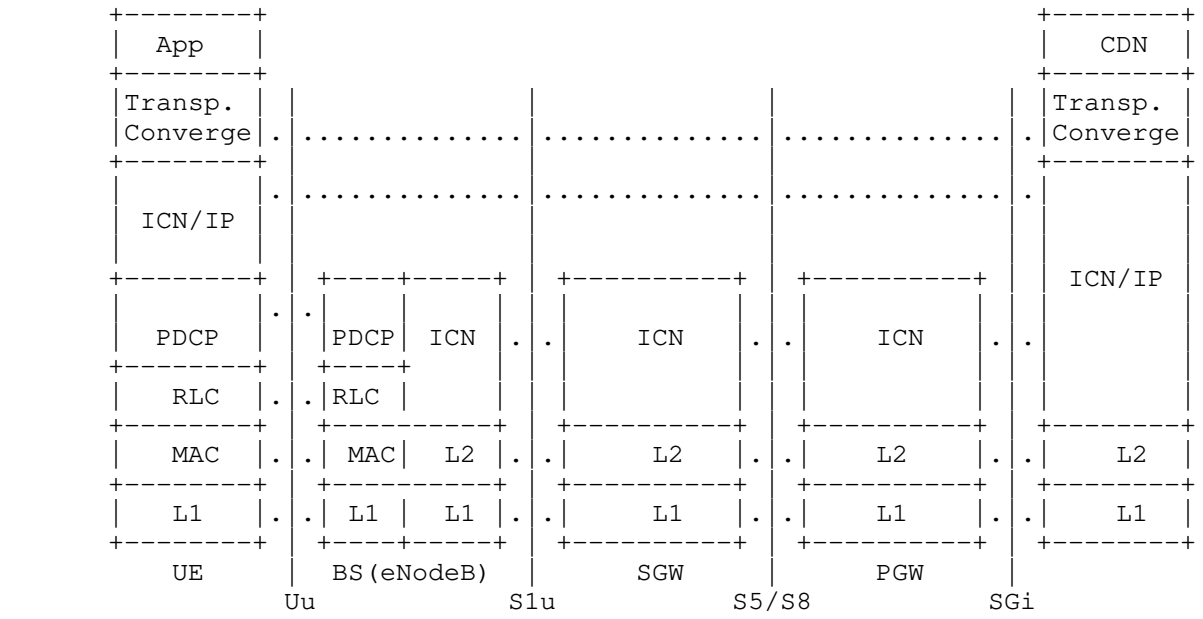


Figure 7: Native ICN Deployment in UE

5.5. ICN Deployment in eNodeB

The eNodeB is a physical point of attachment for the UE, where radio protocols are converted into IP transport protocol for dual stack/ overlay and native ICN, respectively (see Figure 6 and Figure 7). When a UE performs an attach procedure, it is assigned an identity either as IP or dual stack (IP and ICN), or ICN. UE can initiate data traffic using any of the following options:

1. Native IP (IPv4 or IPv6)

2. Native ICN

3. Dual stack IP (IPv4/IPv6) or ICN

The UE encapsulates a user data transport request into PDCP layer and sends the information on the air interface to eNodeB, which in turn receives the information and, using PDCP [TS36.323], de-encapsulates

the air-interface messages and converts them to forward to core mobile gateways (SGW, PGW). As shown in Figure 8, to support ICN natively in eNodeB, it is proposed to provide transport convergence layer (TCL) capabilities in eNodeB (similar to as provided in UE), which provides the following functions:

1. It decides the forwarding strategy for a user data request coming from UE. The strategy can decide based on preference indicated by the application, such as congestion, cost, QoS, and so on.

2. eNodeB to provide open Application Programming Interface (API) to external management systems, to provide capability to eNodeB to program the forwarding strategies.

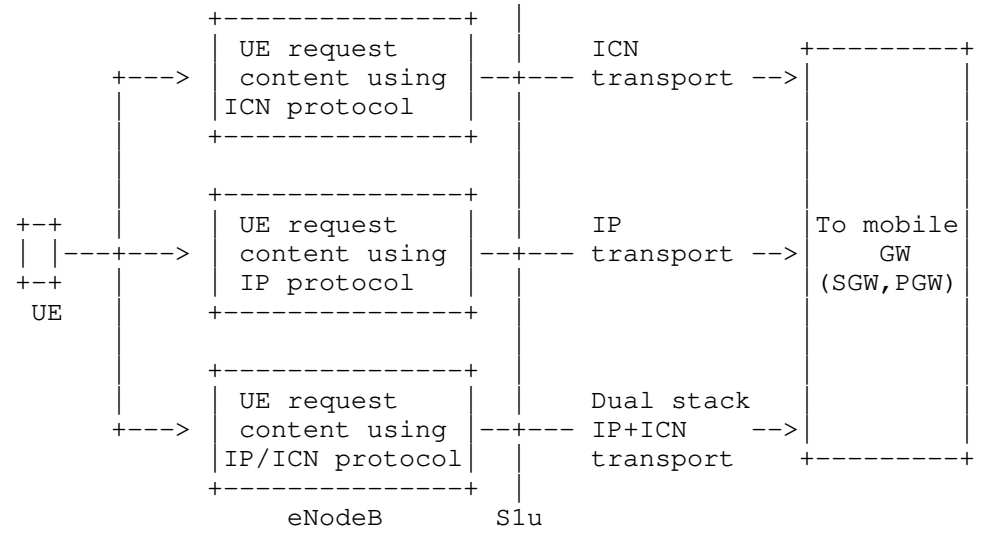


Figure 8: Native ICN Deployment in eNodeB

3. eNodeB can be upgraded to support three different types of transport: IP, ICN, and dual stack IP+ICN towards mobile gateways, as depicted in Figure 8. It is also proposed to deploy IP and/or ICN forwarding capabilities into eNodeB, for efficient transfer of data between eNodeB and mobile gateways. Following are choices for forwarding a data request towards mobile gateways:

    1. Assuming eNodeB is IP enabled and the UE requests an IP transfer, eNodeB forwards data over IP.

2. Assuming eNodeB is ICN enabled and the UE requests an ICN transfer, eNodeB forwards data over ICN.

3. Assuming eNodeB is IP enabled and the UE requests an ICN transfer, eNodeB overlays ICN on IP and forwards user plane traffic over IP.

4. Assuming eNodeB is ICN enabled and the UE requests an IP transfer, eNodeB overlays IP on ICN and forwards user plane traffic over ICN [IPoICN].

5.6. ICN Deployment in Packet Core (SGW, PGW) Gateways

Mobile gateways---also known as Evolved Packet Core (EPC)--include SGW, PGW, which perform session management for UE from the initial attach to disconnection. When UE is powered on, it performs NAS signaling and attaches to PGW after successful authentication. PGW is an anchoring point for UE and responsible for service creations, authorization, maintenance, and so on. The Entire functionality is managed using IP address(es) for UE.

To implement ICN in EPC, the following functions are proposed:

1. Insert ICN attributes in session management layer as additional functionality with IP stack. Session management layer is used for performing attach procedures and assigning logical identity to user. After successful authentication by HSS, MME sends a create session request (CSR) to SGW and SGW to PGW.

2. When MME sends Create Session Request message (Step 12 in [TS23.401]) to SGW or PGW, it includes a Protocol Configuration Option Information Element (PCO IE) containing UE capabilities. We can use PCO IE to carry ICN-related capabilities information from UE to PGW. This information is received from UE during the initial attach request in MME. Details of available TLV, which can be used for ICN, are given in subsequent sections. UE can support either native IP, ICN+IP, or native ICN. IP is referred to as both IPv4 and IPv6 protocols.

3. For ICN+IP-capable UE, PGW assigns the UE both an IP address and ICN identity. UE selects either of the identities during the initial attach procedures and registers with the network for session management. For ICN-capable UE, it will provide only ICN attachment. For native IP-capable UE, there is no change.

4. To support ICN-capable attach procedures and use ICN for user plane traffic, PGW needs to have full ICN protocol stack functionalities. Typical ICN capabilities include functions such

as content store (CS), Pending Interest Table (PIT), Forwarding Information Base (FIB) capabilities, and so on. If UE requests ICN in PCO IE, then PGW registers UE with ICN names. For ICN forwarding, PGW caches content locally using CS functionality.

5. PCO IE described in [TS24.008] (see Figure 10.5.136 on page 598) and [TS24.008] (see Table 10.5.154 on page 599) provide details for different fields.

    1. Octet 3 (configuration protocols define PDN types), which contains details about IPv4, IPv6, both or ICN.

    2. Any combination of Octet 4 to Z can be used to provide additional information related to ICN capability. It is most important that PCO IE parameters are matched between UE and mobile gateways (SGW, PGW) so they can be interpreted properly and the UE can attach successfully.

6. Deployment of ICN functionalities in SGW and PGW should be matched with UE and eNodeB because they will exchange ICN protocols and parameters.

7. Mobile gateways SGW, PGW will also need ICN forwarding and caching capability. This is especially important if CUPS is implemented. User Plane Function (UPF), comprising the SGW and PGW user plane, will be located at the edge of the network and close to the end user. ICN-enabled gateway means that this UPF would serve as a forwarder and should be capable of caching, as is the case with any other ICN-enabled node.

8. The transport between PGW and CDN provider can be either IP or ICN. When UE is attached to PGW with ICN identity and communicates with an ICN-enabled CDN provider, it will use ICN primitives to fetch the data. On the other hand, for a UE attached with an ICN identity, if PGW must communicate with an IP enabled CDN provider, it will have to use an ICN-IP interworking gateway to perform conversion between ICN and IP primitives for data retrieval. In the case of CUPS implementation with an offload close to the edge, this interworking gateway can be collocated with the UPF at the offload site to maximize the path optimization. Further study is required to understand how this ICN-to-IP (and vice versa) interworking gateway would function.

6. Security and Privacy Considerations

This section will cover some security and privacy considerations in user equipment (UE) and LTE network because of introduction of ICN.

6.1. Security Considerations

To ensure only authenticated UEs are connected to the network, LTE mobile network implements various security mechanisms. From the perspective of ICN deployment in the user plane, it needs to take care of the following security aspects:

1. UE authentication and authorization
2. Radio or air interface security
3. Denial of service attacks on the mobile gateway, services either by the UE or by external entities in the Internet
4. Content poisoning either in transport or servers
5. Content cache pollution attacks
6. Secure naming, routing, and forwarding
7. Application security

Security over the LTE air interface is provided through cryptographic techniques. When UE is powered up, it performs a key exchange between UE's USIM and HSS/Authentication Center using NAS messages, including ciphering and integrity protections between UE and MME. Details for secure UE authentication, key exchange, ciphering, and integrity protections messages are given in the 3GPP call flow [TS23.401]. With ICN we are modifying protocol stack for user plane and not control plane. The NAS signaling is exchanged between UE and mobile gateways e.g. MME, using control plane, therefore there is no adverse impact of ICN on UE.

LTE uses IP transport in its mobile backhaul (between eNodeB and core network). In case of provider-owned backhaul, it may not be necessary to implement any security mechanisms because the entire IP transport is owned by service provider. Deployment of security gateways and encryption might be necessary when IP transport is provided by other provider as shared media or leased lines. The native IP transport continues to leverage security mechanism such as Internet key exchange (IKE) and the IP security protocol (IPsec). More details of mobile backhaul security are provided in 3GPP network security specifications [TS33.310] and [TS33.320]. When mobile backhaul is upgraded to support dual stack (IP+ICN) or native ICN, it is required to implement security techniques that are deployed in the mobile backhaul. When ICN forwarding is enabled on mobile transport routers, we need to deploy security practices based on [RFC7476] and [RFC7927].

LTE mobile gateways (SGW, PGW) perform some of key functions such as content based online/offline billing and accounting, deep packet inspection (DPI), and lawful interception (LI). When ICN is deployed in user plane , we need to integrate ICN security for sessions between UE and gateway. If we encrypt user plane payload metadata then it might be difficult to perform routing based on contents and it may not work because we need decryption keys at every forwarder to route the content. The content itself can be encrypted between publisher and consumer to ensure privacy. Only the user with right decryption key shall be able to access the content. We need further research for ICN impact on LI, online/offline charging and accounting.

6.2. Privacy Considerations

In any network, caching implies a trade-off between network efficiency and privacy. The activity of users is exposed to the scrutiny of cache owners with whom they may not have any relationship. By monitoring the cache transactions, an attacker could obtain significant information related to the objects accessed, topology and timing of the requests [RFC7945]. Privacy concerns are amplified by the introduction of new network functions such as Information lookup and Network storage, and different forms of communication [FOTIOU]. Privacy risks in ICN can be broadly divided in the following categories [TOURANI]:

1. Timing attack

2. Communication monitoring attack

3. Censorship and anonymity attack

4. Protocol attack

5. Naming-signature privacy

Introduction of TCL effectively enables ICN at the application and/or transport layer, depending on the scenario described in section 5. Enabling ICN in LTE networks is expected to increase efficiency by taking advantage of ICN’s inherent characteristics. While this approach would potentially leave some of the above-mentioned privacy concerns open, a mere presence of the TCL does not present increased risk and vulnerability.

1. IPoIP Section 5.2 would not be affected as TCL has no role in it and ICN does not apply

2. ICNoICN scenario Section 5.2 has increased risk of a privacy attack, and that risk is applicable to ICN protocol in general rather than specifically to the LTE implementation. Since this scenario describes communication over ICN transport, every forwarder in the path could be a potential risk for privacy attack

3. ICNoIP scenario Section 5.2 uses IP for transport, so the only additional ICN-related potential privacy risk areas are the endpoints (consumer and publisher) where, at the application layer, content is being served

4. IPoICN scenario Section 5.2 could have potentially increased risk due to possible vulnerability of the forwarders in the path of ICN transport

As shown above, introduction of TCL as a vehicle to implement ICN in LTE does not present additional privacy risk beyond issues already identified as they apply to ICN in general. Further research in this area is needed.

7. Summary

In this draft, we have discussed complexities of LTE network and key dependencies for deploying ICN in user plane data transport. Different deployment options described cover aspects such as inter-operability and multi-technology, which is a reality for any Service Provider. One can use LTE gateway software and ICN simulator and deploy ICN data transport in user plane as an overlay, dual stack (IP + ICN), hICN, or natively (by integrating ICN with CDN, eNodeB, SGW, PGW and transport network). Notice that, for deployment scenarios discussed above, additional study is required for lawful interception, billing/mediation, network slicing, and provisioning APIs.

Edge Computing [CHENG] provides capabilities to deploy functionalities such as Content Delivery Network (CDN) caching and mobile user plane functions (UPF) [TS23.501]. Recent research for delivering real-time video content [MPVCICN] using ICN has also been proven to be efficient [NDNRTC] and can be used towards realizing the benefits of ICN deployment in eNodeB, edge computing, mobile gateways (SGW, PGW) and CDN. The key aspect for ICN is in its seamless integration in LTE and 5G networks with tangible benefits so we can optimize content delivery using a simple and scalable architecture. The authors will continue to explore how ICN forwarding in edge computing could be used for efficient data delivery from the mobile edge.

Based on our study of control plane signaling, it is not beneficial to deploy ICN with existing protocols unless further changes are introduced in the control protocol stack itself. As mentioned in [TS23.501], the 5G network architecture proposes a simplification of control plane messages and can be a candidate for the use of ICN.

As a starting step towards ICN user plane deployment, it is proposed to incorporate protocol changes in UE, eNodeB, SGW/PGW for data transport. ICN has inherent capabilities for mobility and content caching, which can improve the efficiency of data transport for unicast and multicast delivery. The authors welcome contributions and suggestions, including those related to further validations of the principles by implementing prototype and/or proof of concept in the lab and in the production environment.

8. Acknowledgements

We thank all contributors, reviewers, and the chairs for the valuable time in providing comments and feedback that helped improve this draft. We specially want to mention the following members of the IRTF Information-Centric Networking Research Group (ICNRG), listed in alphabetical order: Thomas Jagodits, Luca Muscariello, David R. Oran, Akbar Rahman, Martin J. Reed, and Thomas C. Schmidt.

The IRSG review was provided by Colin Perkins.

9. References

9.1. Normative References

[TS24.008]
3GPP, "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3", 3GPP TS 24.008 3.20.0, December 2005, <http://www.3gpp.org/ftp/Specs/html-info/24008.htm>.

[TS25.323]
3GPP, "Packet Data Convergence Protocol (PDCP) specification", 3GPP TS 25.323 3.10.0, September 2002, <http://www.3gpp.org/ftp/Specs/html-info/25323.htm>.

[TS29.274]
3GPP, "3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunneling Protocol for Control plane (GTPv2-C); Stage 3", 3GPP TS 29.274 10.11.0, June 2013, <http://www.3gpp.org/ftp/Specs/html-info/29274.htm>.

[TS29.281]
3GPP, "General Packet Radio System (GPRS) Tunneling Protocol User Plane (GTPv1-U)", 3GPP TS 29.281 10.3.0, September 2011, <http://www.3gpp.org/ftp/Specs/html-info/29281.htm>.

[TS36.323]
3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification", 3GPP TS 36.323 10.2.0, January 2013, <http://www.3gpp.org/ftp/Specs/html-info/36323.htm>.

9.2. Informative References

[ALM]
Auge, J., Carofiglio, G., Grassi, G., Muscariello, L., Pau, G., and X. Zeng, "Anchor-Less Producer Mobility in ICN", Proceedings of the 2Nd ACM Conference on Information-Centric Networking, ACM-ICN'15, ACM DL, pp.189-190, September 2013, <https://dl.acm.org/citation.cfm?id=2812601>.

[BROWER]
Brower, E., Jeffress, L., Pezeshki, J., Jasani, R., and E. Ertekin, "Integrating Header Compression with IPsec", MILCOM 2006 - 2006 IEEE Military Communications conference IEEE Xplore DL, pp.1-6, October 2006, <https://ieeexplore.ieee.org/document/4086687>.

[CCN]
"Content Centric Networking", <http://www.ccnx.org>.

[CHENG]
Liang, C., Yu, R., and X. Zhang, "Information-centric network function virtualization over 5g mobile wireless networks", IEEE Network Journal vol. 29, number 3, pp. 68-74, June 2015, <https://ieeexplore.ieee.org/document/7113228>.

[EPCCUPS]
Schmitt, P., Landais, B., and F. Yong Yang, "Control and User Plane Separation of EPC nodes (CUPS)", 3GPP The Mobile Broadband Standard, July 2017, <http://www.3gpp.org/news-events/3gpp-news/1882-cups>.

[FOTIOU]
Fotiou, N. and G. Polyzos, "ICN privacy and name based security", ACM-ICN '14: Proceedings of the 1st ACM Conference on Information-Centric Networking ACM Digitial Library, pp. 5-6, September 2014, <https://dl.acm.org/doi/10.1145/2660129.2666711>.

[GALIS] Galis, A., Makhijani, K., Yu, D., and B. Liu, "Autonomic Slice Networking", draft-galis-anima-autonomic-slice-networking-05 (work in progress), September 2018.

[GRAYSON] Grayson, M., Shatzkamer, M., and S. Wainner, "Cisco Press book "IP Design for Mobile Networks"", Cisco Press Networking Technology series, June 2009, <http://www.ciscopress.com/store/ip-design-for-mobile-networks-9781587058264>.

[HICN] Muscariello, L., Carofiglio, G., Auge, J., and M. Papalini, "Hybrid Information-Centric Networking", draft-muscariello-intarea-hicn-04 (work in progress), May 2020.

[ICN5G] Ravindran, R., suthar, P., Trossen, D., and G. White, "Enabling ICN in 3GPP's 5G NextGen Core Architecture", draft-ravi-icnrg-5gc-icn-03 (work in progress), July 2020.

[ICNLOWPAN] Gundogan, C., Schmidt, T., Waehlisch, M., Scherb, C., Marxer, C., and C. Tschudin, "ICN Adaptation to LowPAN Networks (ICN LoWPAN)", draft-irtf-icnrg-icnlowpan-08 (work in progress), May 2020.

[ICNQoS] Al-Naday, M., Bontozoglou, A., Vassilakis, G., and M. Reed, "Quality of Service in an Information-Centric Network", 2014 IEEE Global Communications Conference IEEE Xplore DL, pp. 1861-1866, December 2014, <https://ieeexplore.ieee.org/document/7037079>.

[IPoICN] Trossen, D., Read, M., Riihijarvi, J., Georgiades, M., Fotiou, N., and G. Xylomenos, "IP over ICN - The better IP?", 2015 European Conference on Networks and Communications (EuCNC) IEEE Xplore DL, pp. 413-417, June 2015, <https://ieeexplore.ieee.org/document/7194109>.

[MBICN] Carofiglio, G., Gallo, M., Muscariello, L., and D. Perino, "Scalable mobile backhauling via information-centric networking", The 21st IEEE International Workshop on Local and Metropolitan Area Networks, Beijing, pp. 1-6, April 2015, <https://ieeexplore.ieee.org/document/7114719>.

[MECSPEC] "Mobile Edge Computing (MEC); Framework and Reference Architecture", ETSI European Telecommunication Standards Institute (ETSI) MEC specification, March 2016, <https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf>.

[MPVCICN] Jangam, A., Ravindran, R., Chakraborti, A., Wan, X., and G. Wang, "Realtime multi-party video conferencing service over information centric network", IEEE International Conference on Multimedia and Expo Workshops (ICMEW) Turin, Italy, pp. 1-6, June 2015, <https://ieeexplore.ieee.org/document/7169810>.

[NDNRTC] Gusev, P., Wang, Z., Burke, J., Zhang, L., Yoneda, T., Ohnishi, R., and E. Muramoto, "Real-time Streaming Data Delivery over Named Data Networking,", IEICE Transactions on Communications vol. E99.B, pp. 974-991, May 2016, <https://doi.org/10.1587/transcom.2015AMI0002>.

[NGMN] Robson, J., "Backhaul Provisioning for LTE-Advanced and Small Cells", Next Generation Mobile Networks, LTE-Advanced Transport Provisioning, V0.0.14, October 2015, <https://www.ngmn.org/wp-content/uploads/Publications/2015/150929_NGMN_P-SmallCells_Backhaul_for_LTE-Advanced_and_Small_Cells.pdf>.

[OFFLOAD] Rebecchi, F., Dias de Amorim, M., Conan, V., Passarella, A., Bruno, R., and M. Conti, "Data Offloading Techniques in Cellular Networks: A Survey", IEEE Communications Surveys and Tutorials, IEEE Xplore DL, vol:17, issue:2, pp.580-603, November 2014, <https://ieeexplore.ieee.org/document/6953022>.

[OLTEANU] Olteanu, A. and P. Xiao, "Fragmentation and AES Encryption Overhead in Very High-speed Wireless LANs", Proceedings of the 2009 IEEE International Conference on Communications ICC'09, ACM DL, pp.575-579, June 2009, <http://dl.acm.org/citation.cfm?id=1817271.1817379>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <https://www.rfc-editor.org/info/rfc4594>.

[RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <https://www.rfc-editor.org/info/rfc6459>.

[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. Waehlisch, "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>.

[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>.

[SDN5G] Page, J. and J. Dricot, "Software-defined networking for low-latency 5G core network", 2016 International Conference on Military Communications and Information Systems (ICMCIS) IEEE Xplore DL, pp. 1-7, May 2016, <https://ieeexplore.ieee.org/document/7496561>.

[TLVCOMP] Mosko, M., "Header Compression for TLV-based Packets", ICNRG Buenos Aires IETF 95, April 2016, <https://datatracker.ietf.org/meeting/interim-2016-icnrg-02/materials/slides-interim-2016-icnrg-2-7>.

[TOURANI] Tourani, R., Misra, S., Mick, T., and G. Panwar, "Security, Privacy, and Access Control in Information-Centric Networking: A Survey", IEEE Communications Surveys and Tutorials Volume 20, Issue 1, pp 566-600, September 2017, <https://ieeexplore.ieee.org/document/8027034>.

[TS23.203]
3GPP, "Policy and charging control architecture", 3GPP TS 23.203 10.9.0, September 2013, <http://www.3gpp.org/ftp/Specs/html-info/23203.htm>.

[TS23.401]
3GPP, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access", 3GPP TS 23.401 10.10.0, March 2013, <http://www.3gpp.org/ftp/Specs/html-info/23401.htm>.

[TS23.501]
3GPP, "System Architecture for the 5G System", 3GPP TS 23.501 15.2.0, June 2018, <http://www.3gpp.org/ftp/Specs/html-info/23501.htm>.

[TS23.714]
3GPP, "Technical Specification Group Services and System Aspects: Study on control and user plane separation of EPC nodes", 3GPP TS 23.714 0.2.2, June 2016, <http://www.3gpp.org/ftp/Specs/html-info/23714.htm>.

[TS29.060]
3GPP, "General Packet Radio Service (GPRS); GPRS Tunneling Protocol (GTP) across the Gn and Gp interface", 3GPP TS 29.060 3.19.0, March 2004, <http://www.3gpp.org/ftp/Specs/html-info/29060.htm>.

[TS33.310]
3GPP, "Network Domain Security (NDS); Authentication Framework (AF)", 3GPP TS 33.310 10.7.0, December 2012, <http://www.3gpp.org/ftp/Specs/html-info/33310.htm>.

[TS33.320]
3GPP, "Security of Home Node B (HNB) / Home evolved Node B (HeNB)", 3GPP TS 33.320 10.5.0, June 2012, <http://www.3gpp.org/ftp/Specs/html-info/33320.htm>.

Authors' Addresses

Prakash Suthar
Cisco Systems Inc.
Rosemont, Illinois 60018
USA

Email: psuthar@cisco.com

Milan Stolic
Cisco Systems Inc.
Rosemont, Illinois 60018
USA

Email: mistolic@cisco.com

Anil Jangam (editor)
Cisco Systems Inc.
San Jose, California 95134
USA

Email: anjangam@cisco.com

Dirk Trossen
Huawei Technologies
Riesstrasse 25
Munich 80992
Germany

Email: dirk.trossen@huawei.com

Ravishankar Ravindran
Sterlite Technologies
5201 Greatamerica Pkwy
Santa Clara, California 95054
USA

Email: ravishankar.ravindran@sterlite.com

ICNRG
Internet-Draft
Intended status: Informational
Expires: December 2, 2019

R. Ravindran
Futurewei
P. Suthar
Cisco
D. Trossen
C. Wang
InterDigital Inc.
G. White
CableLabs

May 31, 2019

Enabling ICN in 3GPP's 5G NextGen Core Architecture
draft-ravi-icnrg-5gc-icn-04

Abstract

The proposed 3GPP's 5G core nextgen architecture (5GC) offers flexibility to introduce new user and control plane function, considering the support for network slicing functions, that allows greater flexibility to handle heterogeneous devices and applications. In this draft, we provide a short description of the proposed 5GC architecture, including recent efforts to provide cellular Local Area Network (LAN) connectivity, followed by extensions to 5GC's control and user plane to support Packet Data Unit (PDU) sessions from Information-Centric Networks (ICN). In addition, ICN over 5GLAN is also described.

Table of Contents

1. Introduction

The objective of this draft is to propose an architecture to enable information-centric networking (ICN) in the proposed 5G Next-generation Core network architecture (5GC) by leveraging its flexibility to allow new user and associated control plane functions. The reference architectural discussions in the 5G core network 3GPP specifications [TS23.501][TS23.502] form the basis of our

discussions. This draft also complements the discussions 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.

Though ICN is a general networking technology, it would benefit 5G particularly from the perspective of mobile edge computing (MEC). The following ICN features shall benefit MEC deployments in 5G:

- 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.

- 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 realtime and non-realtime 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.

- Session Mobility: Existing long-term evolution (LTE) deployments handle session mobility using centralized routing using the MME function, IP anchor points at Packet Data Network Gateway (PDN-GW) and service anchor point called Access Point Name (APN) functionality hosted in PDN-GW. LTE uses tunnel between radio edge (eNodeB) and PDN-GW for each mobile device attached to network. This design fails when service instances are replicated close to radio access network (RAN) instances, requiring new techniques to handle session mobility. In contrast, application-bound identifier and name resolution split principle considered for the ICN is shown to handle host mobility quite efficiently [ICNMOB].

In this document, we first discuss 5GC’s design principals that allows the support of new network architectures. Then we summarize

the 5GC proposal, followed by control and user plane extensions required to support ICN PDU sessions. This is followed by discussions on enabling IP over ICN over 3GPP proposed 5GLAN service framework. We then discuss deployment considerations for both ICN over 5GC and IP over ICN over 5GLAN.

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].

5G-New Radio (5G-NR): This refers to the new radio access interface developed to support 5G wireless interface [TS38.300].

User Plane Function (UPF): UPF is the generalized logical data plane function with context of the UE PDU session. UPFs can play many role, such as, being an flow classifier (UL-CL) (defined next), a PDU session anchoring point, or a branching point.

Uplink Classifier (UL-CL): This is a functionality supported by an UPF that aims at diverting traffic (locally) to local data networks based on traffic matching filters applied to the UE traffic.

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): Manages 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 CUPS (discussed in the next section) 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.

5GLAN Service: A service over the 5G system offering private communication using IP and/or non-IP type communications.

3. 5G NextGen Core Design Principles

The 5GC architecture is based on the following design principles that allows it to support new service networks like ICN efficiently compared to LTE networks:

- Control and User plane split (CUPS): This design principle moves away from LTE's vertically integrated control/user plane design (i.e., Serving Gateway, S-GW, and Packet Data Network Gateway, P-GW) to one espousing an NFV framework with network functions separated from the hardware for service-centricity, scalability, flexibility and programmability. In doing so, network functions can be implemented both physically and virtually, while allowing each to be customized and scaled based on their individual requirements, also allowing the realization of multi-slice co-existence. This feature also allows the introduction of new user plane functions (UPF) in 5GC. UPFs can play many roles, such as, being an uplink flow classifier (UL-CL), a PDU session anchor point, a branching point function, or one based on new network architectures like ICN with new control functions, or re-using/ extending the existing ones to manage the new user plane realizations.

- Decoupling of RAT and Core Network : Unlike LTE's unified control plane for access and the core, 5GC offers control plane separation of the RAN from the core network. This allows the introduction of new radio access technologies (RAT) along with slices based on new network architectures, offering the ability to map heterogeneous

RAN flows to arbitrary core network slices based on service requirements.

- Non-IP PDU Session Support : A PDU session is defined as the logical connection between the UE and the data network (DN). 5GC offers a scope to support both IP and non-IP PDU (termed as "unstructured" payload), and this feature can potentially allow the support for ICN PDUs by extending or re-using the existing control functions. More discussions on taking advantage of this feature in ICN's context is presented in Section 4.2.2.2.

- Service Centric Design: 5GC's service orchestration and control functions, such as naming, addressing, registration/authentication and mobility, will utilize API design similar to those used in cloud technologies. Doing so enables opening up interfaces for authorized service function interaction and creating service level extensions to support new network architectures. These APIs include the well accepted Get/Response and Pub/Sub approaches, while not precluding the use of point-to-point procedural approach among 5GC functional units (where necessary).

- Distributed LAN Support: utilizing the aforementioned unstructured PDU session support, 5GC offers the capability to expose a Layer 2 LAN service to cellular user equipment. Such distributed LAN targets to complement those in fixed broadband, including local WLAN fanouts. Through such LAN capability, services can be realized by being virtually embedded into an intranet deployment with dedicated Internet-facing packet gateway functionality. Examples for such services, among others, are those related to Industrial IoT, smart city services and others. Utilizing this capability for ICN-based services is presented in Section 5.1.

4. 5GC Architecture with ICN Support

4.1. 5G NextGen Core Architecture

In this section, for brevity purposes, we restrict the discussions to the control and user plane functions relevant to an ICN deployment discussion in Section 4.2. More exhaustive discussions on the various architecture functions, such as registration, connection and subscription management, can be found in[TS23.501][TS23.502].

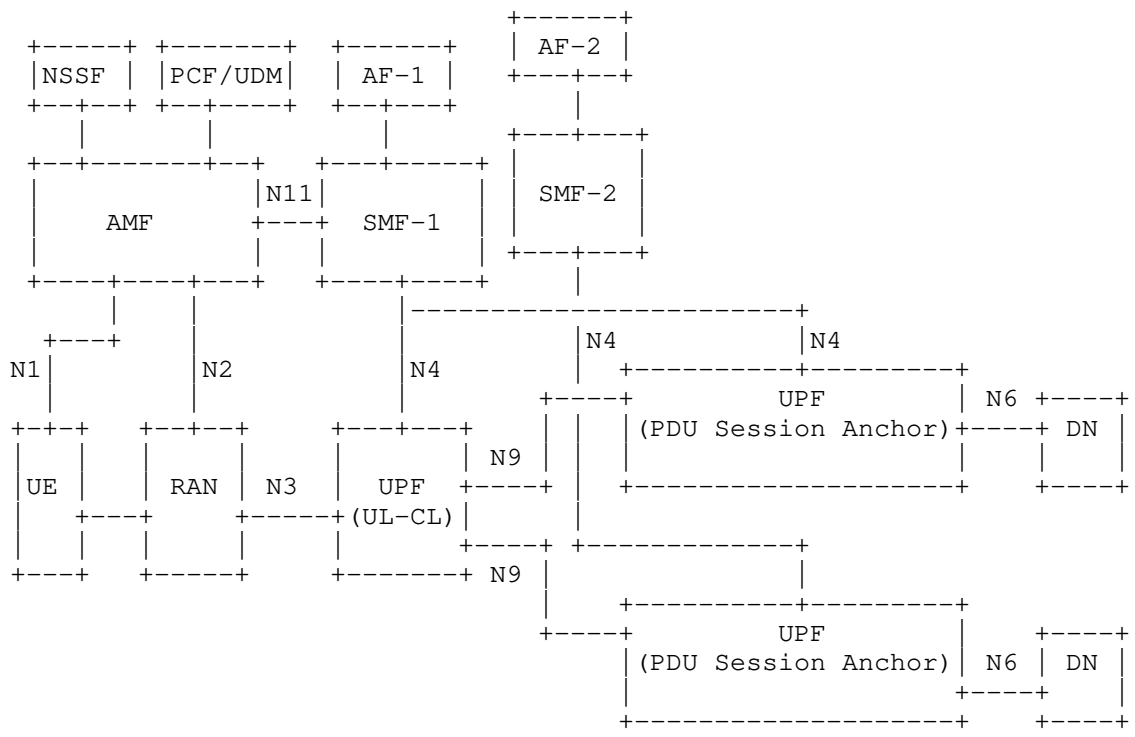


Figure 1: 5G Next Generation Core Architecture

In Figure 1, we show one variant of a 5GC architecture from [TS23.501], for which the functions of UPF's branching point and PDU session anchoring are used to support inter-connection between a UE and the related service or packet data networks (or PDNs) managed by the signaling interactions with control plane functions. In 5GC, control plane functions can be categorized as follows:

- Common control plane functions that are common to all slices and which include the Network Slice Selection Function (NSSF), Policy Control Function (PCF), and Unified Data Management (UDM) among others.

- Shared or slice specific control functions, which include the Access and Mobility Function (AMF), Session and Management Function (SMF) and the Application Function (AF).

AMF serves multiple purposes: (i) device authentication and authorization; (ii) security and integrity protection to non-access stratum (NAS) signaling; (iii) tracking UE registration in the

operator's network and mobility management functions as the UE moves among different RANs, each of which might be using different radio access technologies (RAT).

NSSF handles the selection of a particular slice for the PDU session request from the user entity (UE) using the Network Slice Selection Assistance Information (NSSAI) parameters provided by the UE and the configured user subscription policies in PCF and UDM functions. Compared to LTE's evolved packet core (EPC), where PDU session states in RAN and core are synchronized with respect to management, 5GC decouples this using NSSF by allowing PDU sessions to be defined prior to a PDU session request by a UE (for other differences see [lteversus5g]). This decoupling allows policy based inter-connection of RAN flows with slices provisioned in the core network. This functionality is useful particularly towards new use cases related to M2M and IOT devices requiring pre-provisioned network resources to ensure appropriate SLAs.

SMF is used to handle IP anchor point selection and addressing functionality, management of the user plane state in the UPFs (such as in uplink classifier (UL-CL), IP anchor point and branching point functions) during PDU session establishment, modification and termination, and interaction with RAN to allow PDU session forwarding in uplink/downlink (UL/DL) to the respective DNs. SMF decisions are also influenced by AF to serve application requirements, for e.g., actions related to introducing edge computing functions.

In the data plane, UE's PDUs are tunneled to the RAN using the 5G RAN protocol[TS38.300]. From the RAN, the PDU's five tuple header information (IP source/destination, port, protocol etc.) is used to map the flow to an appropriate tunnel from RAN to UPF. Though the current 5GC proposal[TS23.501] follows LTE on using GPRS tunneling protocol (GTP) tunnel from NR to the UPF to carry data PDUs and another one for the control messages to serve the control plane functions; there are ongoing discussions to arrive upon efficient alternatives to GTP.

4.2. ICN over 5GC

In this section, we focus on control and user plane enhancements required to enable ICN within 5GC, and identify the interfaces that require extensions to support ICN PDU sessions. Explicit support for ICN PDU sessions within access and 5GC networks will enable applications to leverage the core ICN features while offering it as a service to 5G users.

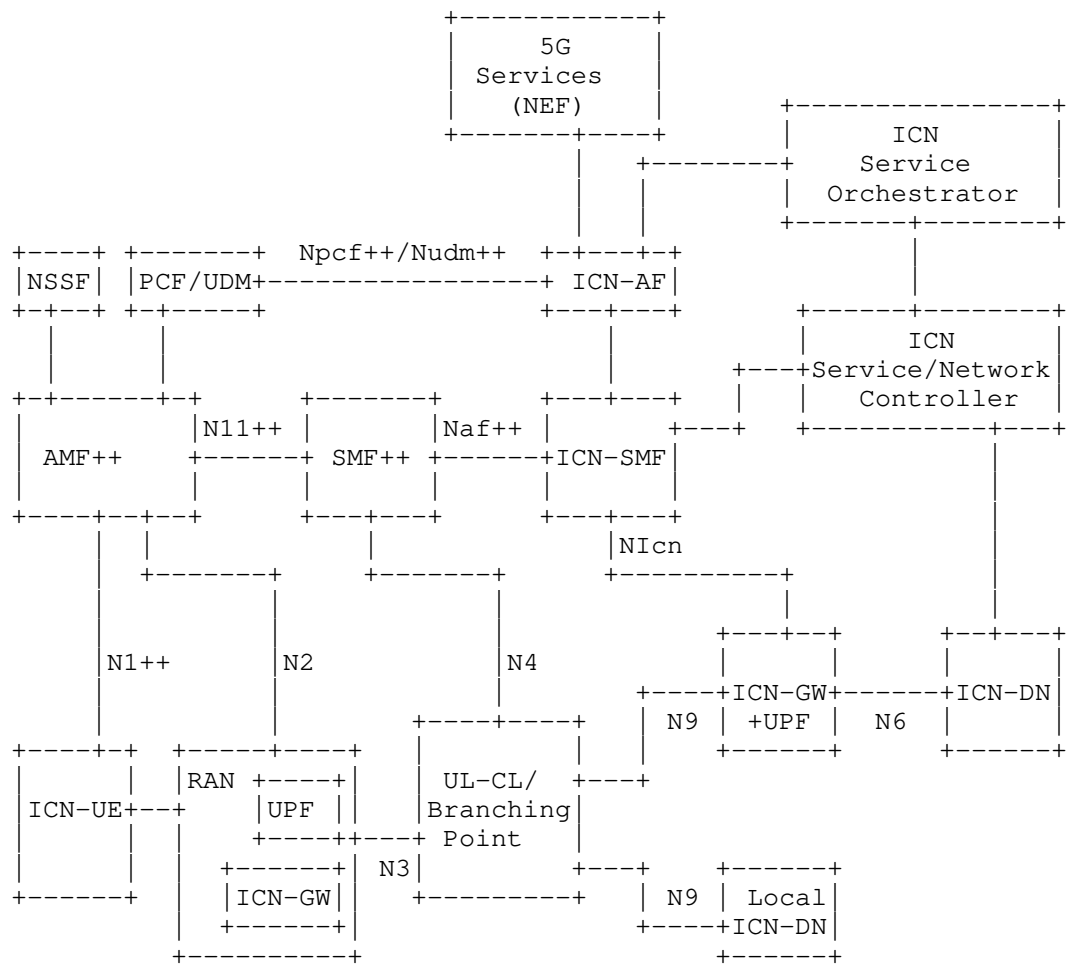


Figure 2: 5G Next Generation Core Architecture with ICN support

For an ICN-enabled 5GC network, the assumption is that the UE may have applications that can run over ICN or IP, for instance, UE's operating system offering applications to operate over ICN [Jacobson] or IP-based networking sockets. There may also be cases where UE is exclusively based on ICN. In either case, we identify an ICN enabled UE as ICN-UE. Different options exist to implement ICN in UE as described in [I-D.irtf-icnrg-icn-lte-4g] which is also applicable for 5G UE to enable formal ICN session handling, such as, using a Transport Convergence Layer (TCL) above 5G-NR, through IP address assignment from 5GC or using 5GC provision of using unstructured PDU session mode during the PDU session establishment process, which is

discussed in Section 4.2.2.2. Such convergence layer would implement necessary IP over ICN mappings, such as those described in [TROSSEN], for IP-based applications that are assigned to be transported over an ICN network. 5G UE can also be non-mobile devices or an IOT device using radio specification which can operate based on [TS38.300].

5GC will take advantage of network slicing function to instantiate heterogeneous slices, the same framework can be extended to create ICN slices as well [Ravindran]. This discussion also borrows ideas from[TS23.799], which offers a wide range of architectural discussions and proposals on enabling slices and managing multiple PDU sessions with local networks (with MEC) and its associated architectural support (in the service, control and data planes) and procedures within the context of 5GC.

Figure 2 shows the proposed ICN-enabled 5GC architecture. In the figure, the new and modified functional components are identified that interconnects an ICN-DN with 5GC. The interfaces and functions that require extensions to enable ICN as a service in 5GC can be identified in the figure with a '++' symbol. We next summarize the control, user plane and normative interface extensions that help with the formal ICN support.

4.2.1. Control Plane Extensions

To support interconnection between ICN UEs and the appropriate ICN DN instances, we consider the following additional control plane extensions to orchestrate ICN services in coordination with 5GC's control components.

- Authentication and Mobility Function (AMF++): ICN applications in the UEs have to be authorized to access ICN DNs. For this purpose, as in [TS23.501], operator enables ICN as a DN offering ICN services. As a network service, ICN-UE should also be subscribed to it and this is imposed using the PCF and UDM, which may interface with the ICN Application Function (ICN-AF) for subscription and session policy management of ICN PDU sessions. To enable ICN stack in the UE, AMF++ function has to be enabled with the capability of authenticating UE's attach request for ICN resources in 5GC. The request can be incorporated in NSSI parameter to request either ICN specific slice or using ICN in existing IP network slice when the UE is dual stacked. AMF++ can potentially be extended to also support ICN specific bootstrapping (such as naming and security) and forwarding functions to configure UE's ICN layer. These functions can also be handled by the ICN-AF and ICN control function in the UE after setting PDU session state in 5GC. Here, the recommendation is not about redefining the 5G UE attach procedures, but to extend the attach

procedures messages to carry ICN capabilities extensions in addition to supporting existing IP based services. The extensions should allow a 5G UE to request authentication to 5GC either in ICN, IP or dual-stack (IP and ICN) modes. Further research is required to optimize 5G attach procedures so that an ICN capable UE can be bootstrapped by minimizing the number of control plane messages. One possibility is to leverage existing 5G UE attach procedures as described in 3GPP's [TS23.502], where the UE can provide ICN identity in the LTE equivalent protocol configuration option information element (PCO-IE) message during the attach request as described in [I-D.irtf-icnrg-icn-lte-4g]. In addition, such requirement can be also be provided by the UE in NSSI parameters during initial attach procedures. Alternately, ICN paradigm offers name-based control plane messaging and security which one can leverage during the 5G UE attach procedures, however this requires further research.

- Session Management Function (SMF++): Once a UE is authenticated to access ICN service in network, SMF manages to connect UE's ICN PDU sessions to the ICN DN in the UL/DL. SMF++ should be capable to manage both IP, ICN or dual stack UE with IP and ICN capabilities. To support ICN sessions, SMF++ creates appropriate PDU session policies in the UPF, which include UL-CL and ICN gateway (ICN-GW) (discussed in Section 4.2.2) through the ICN-SMF. For centrally delivered services, ICN-GW could also multiplex as an IP anchor point for IP applications. If MEC is enabled, these two functions would be distributed, as the UL-CL will re-route the flow to a local ICN-DN. 3GPP has defined IP based session management procedures to handle UE PDU sessions in TS23.502. For ICN UE we can either leverage same procedures when ICN is deployed as an overlay protocol. Towards this, SMF++ interfaces with AMF++ over N11++ to enable ICN specific user plane functions, which include tunnel configuration and traffic filter policy to inter-connect UE with the appropriate radio and the core slice. Furthermore, AMF++ sets appropriate state in the RAN and the UE that directs ICN flows to the chosen ICN UL-CL in the core network, and towards the right UE in the downlink.

- ICN Session Management Function (ICN-SMF): ICN-SMF serves as control plane for the ICN state managed in ICN-GW. This function can be either incorporated as part of SMF++ or as a stand-alone one. This function interacts with SMF++ to obtain and also push ICN PDU session management information for the creation, modification and deletion of ICN PDU sessions in ICN-GW. For instance, when new ICN slices are provisioned by the ICN service orchestrator, ICN-SMF requests a new PDU session to the SMF that extends to the RAN. While SMF++ manages the tunnels to interconnect ICN-GW to UL-CL, ICN-SMF creates the appropriate

forwarding state in ICN-GW (using the forwarding information base or FIB) to enable ICN flows over appropriate tunnel interfaces managed by the SMF++. In addition, it also signals resource management rules to share compute, bandwidth, storage/cache resources among multiple slice instances co-located in the ICN-GW.

- ICN Application Function (ICN-AF): ICN-AF represents the application controller function that interfaces with ICN-SMF and PCF/UDM function in 5GC. In addition to transferring ICN forwarding rules to ICN-SMF, ICN-AF also interfaces with PCF/UDM to transfer user profile and subscription policies along with session management requirement to UE's ICN PDU session in the 5GC network. ICN-AF is an extension of the ICN service orchestration function, which can influence both ICN-SMF and in-directly SMF++ to steer traffic based on ICN service requirements. ICN-AF can also interact with the northbound 5G operator's service functions, such as network exposure function(NEF) that exposes network capabilities, for e.g. location based services, that can be used by ICN-AF for proactive ICN PDU session and slice management and offer additional capabilities to the ICN slices.

4.2.1.1. Normative Interface Extensions

- N1++/N11++: This extension enables ICN specific control functions to support ICN authentication, configuration and programmability of an ICN-UE via AMF++ and SMF++, and also impose QoS requirements, handle mobility management of an ICN PDU session in 5GC based on service requirements.

- N4: Though this signaling is service agnostic, as discussed in Section 4.2.2, future extensions may include signaling to enable ICN user plane features in these network functions. The extension of N4 to RAN is to handle the case when UPF function collocates with the RAN instance to enable localized ICN DNs.

- NIcn: This extension shall support two functions: (i) control plane programmability to enable ICN PDU sessions applicable to 5GC to map to name based forwarding rules in ICN-GW; (ii)control plane extensions to enable ICN mobility anchoring at ICN-GW, in which case it also acts as POA for ICN flows. Features such as ICN mobility as a service can be supported with this extension [ICNMOB].

- Naf++: This extension supports 5GC control functions such as naming, addressing, mobility, and tunnel management for ICN PDU sessions to interact with SMF++ and AMF++.

- Npcf++/Nudm++: This extension creates an interface to push ICN service and PDU session requirements to PCF and UDM functions that interact with the ICN-AF function for ICN slice specific configuration. These requirements are enforced at various steps, for instance, during ICN application registration, authentication, slice mapping, and provisioning of resources for these PDU sessions in the UPF.

4.2.2. User Plane Extensions

The interconnection of a UE to an ICN-DN comprises of two segments, one from RAN to UL-CL and the other from UL-CL to ICN-GW. These segments use IP tunneling constructs, where the service semantic check at UL-CL is performed using IP's five tuples to determine both UL and DL tunnel mappings. We summarize the relevant UPFs and the interfaces for handling ICN PDU sessions as follows.

- ICN Gateway (ICN-GW): ICN-GW is where the 5GC PDU sessions terminate and ICN service network begins. Compared to the traditional anchor points as in PGW, the ICN-GW is also a service gateway as it can host services or cache content enabled through the ICN architecture. The ICN-GW also includes the UPF functions to manage multiple tunnel interfaces enabling the relay of ICN PDU flows to appropriate UL-CL instances in the DL. Note that there may be multiple ICN-GWs serving different ICN services or slices. ICN-GW also manages other ICN functions such as enforcing the dynamic name based forwarding state, mobility state, in-network service function management, resource management with respect to sharing caching, storage, and compute resources among multiple services[Ravindran].

- ICN Packet Data Network (ICN-(P)DN): ICN-DN represents a set of ICN nodes used for ICN networking and with heterogeneous service resources such as storage and computing points. An ICN network enables both network and application services, with network services including caching, mobility, multicast, multi-path routing (and possibly network layer computing), and application services including network resources (such as cache, storage, network state resources) dedicated to the application.

  * Considering multiple ICN architecture proposals and multiple ICN deployment models discussed in [I-D.irtf-icnrg-deployment-guidelines], an alternate backward compatible (IP-over-)ICN solution is proposed in [TROSSEN]. Such an ICN-(P)DN can simply consist of SDN forwarding nodes and a logically centralized path computation entity (PCE), where the PCE is used to determine suitable forwarding identifiers being used for the path-based forwarding in the

SDN-based transport network. In addition, the PCE is responsible for maintaining the appropriate forwarding rules in the SDN switches. For interconnection to IP-based peering networks, a packet gateway is being utilized that mirrors the convergence layer functionality to map incoming ICN traffic back in to outgoing IP traffic and vice versa. This form of deployment would require minimal changes to the 5GC's user and control plane procedures, as the applications on these IP end points are not exposed (or minimally exposed) to any ICN state or configuration.

- Uplink Classifier (UL-CL): UL-CL enables classification of flows based on source or destination IP address and steers the traffic to an appropriate network or service function anchor point. If the ICN-GW is identified based on service IP address associated with the ICN-UE's flows, UL-CL checks the source or destination address to direct traffic to an appropriate ICN-GW. For native ICN UE, ICN shall be deployed over 5G-NR; here, there may not be any IP association. For such packet flows new classification schema shall be required, such as, using 5G-NR protocol extensions to determine the tunnel interface to forward the ICN payload on, towards the next ICN-GW.

4.2.2.1. Normative Interface Extensions

- N3: Though the current architecture supports heterogeneous service PDU handling, future extensions can include user plane interface extensions to offer explicit support to ICN PDU session traffic, for instance, an incremental caching and computing function in RAN or UL-CL to aid with content distribution.

- N9: Extensions to this interface can consider UPFs to enable richer service functions, for instance to aid context processing. In addition extensions to enable ICN specific encapsulation to piggyback ICN specific attributes such as traffic or mobility data between the UPF branching point and the ICN-GW.

- N6: This interface is established between the ICN-GW and the ICN-DN, whose networking elements in this segment can be deployed as an overlay or as a native Layer-3 network.

4.2.2.2. ICN over non-IP PDU

5GC accommodates non-IP PDU support which is defined for Ethernet or any unstructured data[TS23.501]. This feature allows native support of ICN over 5G RAN, with the potential enablement of ICN-GW in the BS itself as shown in Figure 2. Formalizing this feature to recognize ICN PDUs has many considerations:

- Attach Procedures for UE with Non-IP PDN: Assuming a 5GC can support both IP and non-IP PDN, this requires control plane support. In a typical scenario, when UE sends an attach message to 5GC, the type of PDU connection is indicated in the PCO-IE field, for e.g. in this case as being non-IP PDN to invoke related control plane session management tasks. ICN over non-IP PDU session will allow the UE to attach to 5GC without any IP configuration. 5GC attach procedures specified [TS23.501] can be used to support authentication of UE with PDN type set to non-IP, using existing AUSF/UDM functions in coordination with the ICN-AF function discussed earlier if required.

- User Plane for UE with Non-IP PDN: Without any IP tunnel configuration and ICN's default consumer agnostic mode of operation requires ways to identify the ICN-UE in the user plane, this can be enabled by introducing network identifier in the lower layers such as in the PDCP or MAC layer, that can assist for functions such as policy and charging at the BS and related session management tasks. These identifiers can also be used to demultiplex the DL traffic from the ICN-GW in the BS to the respective ICN-UEs. Also, ICN extensions can be incorporated in control plane signaling to identify an ICN-UE device and these parameters can be used by SMF to conduct non-IP routing. The policing and charging functions can be enforced by the UPF function in the BS which obtains the traffic filtering rules from the SMF. To enable flat ICN network from the BS requires distributed policy, charging and legal intercept which requires further research. Further ICN slice multiplexing can be realized by also piggybacking slice-ID (NSSI) along with device ID to differentiate handover to multiple ICN slices at the base station. Inter-working function (IWF) is required if services based on non-IP UE has to transact or communicate with transport, applications functions or other UE based on IP services. This also has implications on how mobility is managed for such PDU sessions.

- Mobility Handling: Considering mobility can be support by ICN, it is inefficient to traverse other intermediate IP networks between the BS and the next ICN hop. This requires ICN PDU to be handled by an ICN instance in the BS itself, in association with UL-CL function local to the BS as shown in Figure 2. Control plane extensions discussed in Section 4.2.1 can be used in tandem with distributed mobility protocols to handle ICN mobility, one such solution for producer mobility is proposed in [ICNMOB]

- Routing Considerations: Flat ICN network realizations also offers the advantage of optimal routing, compared to anchor point based realization in LTE. This also leads to optimal realization of the data plane considering the absence of overhead due to tunneling

while forwarding ICN traffic. However, developing a routing control plane in to handle the ICN PDU sessions either leveraging SMF functions or a distributed realization requires more investigation. In the centralized approach the SMF could interact with ICN-SMF to set the forwarding rules in the ICN-GW in the BS and other ICN-UPFs, however this may also lead to scalability issues if a flat ICN network is to be realized. This also has implications to route the non-IP PDU sessions efficiently to the closest ICN-MEC instance of the service.

- IP over ICN: Native support of ICN in the RAN raises the possibility of leveraging the mobility functions in ICN protocols as a replacement for GTP tunneling in the 5GC, as described in [I-D.white-icnrg-ipoc] and [TROSSEN].

- Mobile Edge Computing: Another significant advantage is with respect to service-centric edge computing at the ICN-GW or other ICN points, either through explicit hosting of service functions[VSER] in ICN or in-network computing based on NFN proposal[NFN]. A certain level of orchestration is required to ensure service interconnection and its placement with appropriate compute resources and inter-connected with bandwidth resources so that the desired SLA is offered.

4.2.3. Dual Stack ICN Deployment

4.2.3.1. 5G User Plane Protocol Stack

It is important to understand that a User Equipment (UE) can be either consumer (receiving content) or publisher (pushing content for other clients). The protocol stack inside mobile device (UE) is complex as it has to support multiple radio connectivity access to gNB(s).

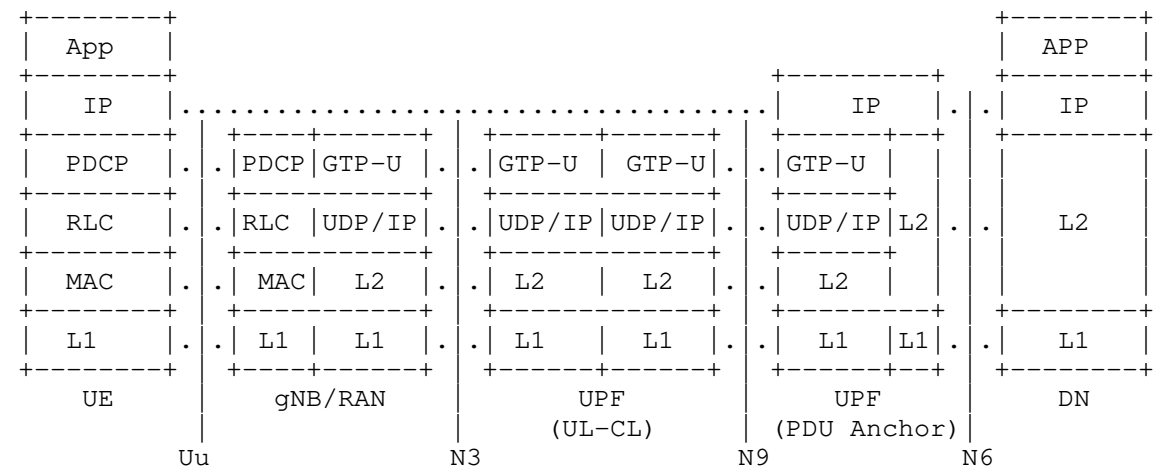


Figure 3: 5G User Plane Protocol Stack

Figure 3 provides high level description of a 5G user plane protocol stack, where: 1) the lower 4 layers (i.e. L1, MAC, RLC, PDCP) at UE is for radio access and air interface to gNB; 2) the IP layer (i.e. PDU layer) at UE is used for providing IP transport infrastructure to support PDU session between UE and UPF (PDU Anchor); 3) GTP-U provides tunneling between gNB and UPF, or between two UPFs. Although UDP/IP exists under GTP-U, IP mainly refers to "IP" between UE and UPF (PDU Anchor) for the rest of this document, unless explicitly clarified; 4) UL-CL is only for uplink traffic and UPF (UL-CL) shall not be needed for downlink traffic towards UE.

4.2.3.2. Protocol Stack for ICN Deployment in 5G

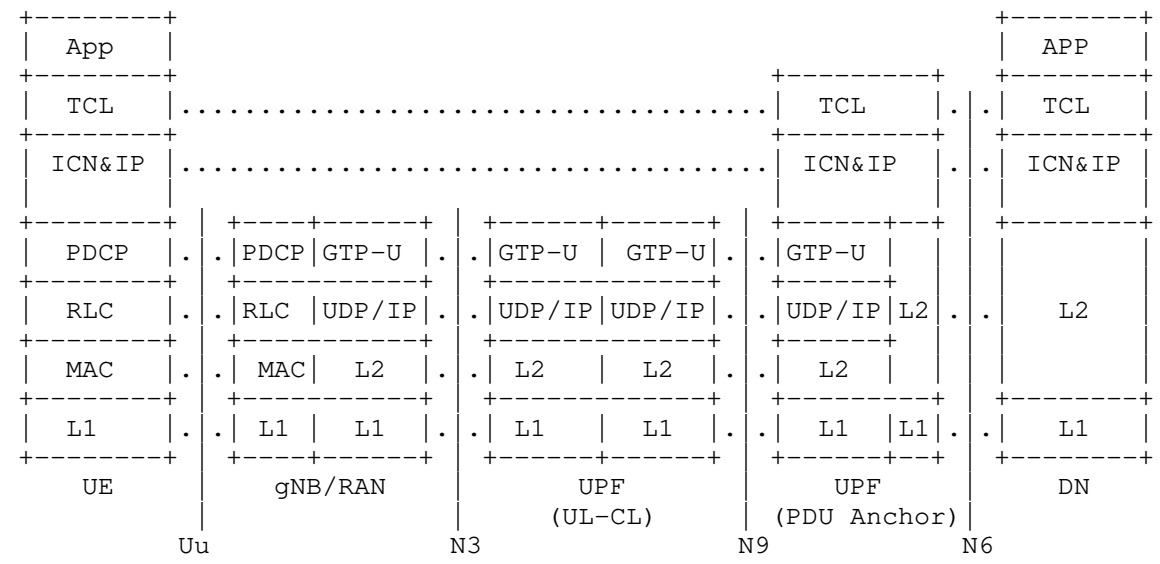


Figure 4: Dual Stack ICN Deployment

ICN can be deployed in dual stack model for 5G user plane as illustrated in Figure 4, where: 1) both ICN and IP (i.e. dual stack) can reside between TCL and PDCP to provide transport infrastructure from UE to UPF (PDU Anchor); 2) in order to support the dual ICN&IP transport layer, PDCP needs some enhancements; 3) a new Transport Convergence Layer (TCL) is introduced to coordinate between applications and ICN&IP transport layer; 4) Applications on top of TCL could be ICN applications or IP applications.

With this dual stack model, four different cases are possible for the deployment of ICN natively and/or with IP dependent on which types of applications (ICN or IP) uses which types of underline transport (ICN or IP), from the perspective of the applications either on UE (or content provider).

Case 1. IP over IP (IPoIP)

In this scenario UE uses applications tightly integrated with the existing IP transport infrastructure. In this option, the TCL has no additional function since the packets are directly forwarded using IP protocol stack, which in turn sends the packets over the IP transport.

Case 2. ICN over ICN (ICNoICN)

Similar to case 1 above, ICN applications tightly integrate with the ICN transport infrastructure. The TCL has no additional responsibility since the packets are directly forwarded using ICN protocol stack, which in turn sends the packets over the ICN transport.

Case 3. ICN over IP (ICNoIP)

In ICN over IP scenario, the underlying IP transport infrastructure is not impacted (i.e., ICN is implemented as an overlay over IP between UE and content provider). IP routing is used from Radio Access Network (gNB) to mobile backhaul, IP core and UPF. UE attaches to UPF (PDU Anchor) using IP address. Content provider in DN is capable of serving content either using IP or ICN, based on UE request.

An alternative approach to implement ICN over IP is provided in Hybrid ICN [I-D.muscariello-intarea-hicn], which implements ICN over IP by mapping of ICN names to the IPv4/IPv6 addresses.

Case 4. IP over ICN (IPoICN)

In IP over ICN scenario, IP application utilize an ICN-based routing while preserving the overall IP protocol semantics, as shown, e.g., in H2020 project [H2020]. Implementing IP services over ICN provides an opportunity leveraging benefit of ICN in the transport infrastructure.

Note that the IP over ICN case could be supported for pure IP (over IP) UEs through introducing a Network Attachment Point (NAP) to interface to an ICN network. Here, the UPF (PDU Anchor) interfaces to said NAP in the northbound; alternatively, the NAP can be integrated as a part of UPF (PDU Anchor). For this scheme, the NAP provides a standard IP network interface towards the IP-enabled UE via UPF (PDU Anchor), encapsulates any received IP service (e.g. HTTP) request into an appropriate ICN packet which is then published as an appropriately formed named information item. Conversely, the NAP subscribes to any appropriately formed named information items, where the information identifier represents any IP-exposed service that is exposed at any IP-level UE locally connected to the NAP. Any received ICN packet is then forwarded to the appropriate local IP-enabled UE after being appropriately decapsulated, recovering the original IP service (e.g. HTTP) request.

In a dual-stack UE that supports the above cases, the TCL helps determine what type of transport (e.g. ICN or IP), as well as type of radio interface (e.g. 5G or WiFi or both), is used to send and receive the traffic based on preference e.g. content location,

content type, content publisher, congestion, cost, quality of service etc. It helps to configure and decide the type of connection as well as the overlay mode (ICNoIP or IPoICN, explained above) between application and the protocol stack (IP or ICN) to be used.

TCL can use a number of mechanisms for the selection of transport (i.e. ICN or IP). It can use a per application configuration through a management interface, possibly even a user-facing setting realized through a user interface, similar to those used today that select cellular over WiFi being used for selected applications. In another option, it might use a software API, which an adapted IP application could use to specify e.g. an ICN transport for obtaining its benefits.

Another potential application of TCL is in implementation of network slicing, where it can have a slice management capability locally or it can interface to an external slice manager through an API [I-D.galis-anima-autonomic-slice-networking]. This solution can enable network slicing for IP and ICN transport selection from the UE itself. The TCL could apply slice settings to direct certain traffic (or applications) over one slice and others over another slice, determined by some form of 'slicing policy'. Slicing policy can be obtained externally from slice manager or configured locally on UE.

4.2.3.3. Protocol Interactions and Impacts

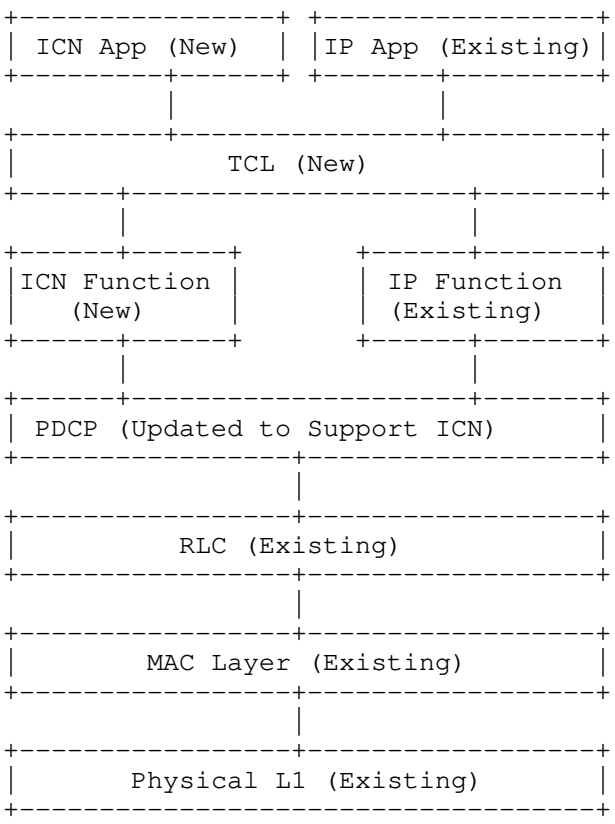


Figure 5: Dual Stack ICN Protocol Interactions at UE

The protocol interactions and impact of supporting tunneling of ICN packet into IP or to support ICN natively are described in Figure 5.

- Existing application layer can be modified to provide options for new ICN based application and existing IP based applications. UE can continue to support existing IP based applications or host new applications developed either to support native ICN as transport, ICNoIP or IPoICN based transport. Application layer has the option of selecting either ICN or IP transport layer as well as radio interface to send and receive data traffic. Our proposal is to provide a common Application Programming Interface (API) to the application developers such that there is no impact on the application development when they choose either ICN or IP transport for exchanging the traffic with the network. TCL function handles the interaction of application with the multiple transport options.

- The TCL helps determine what type of transport (e.g. ICN or IP) as well as type of radio interface (e.g. 5G NR or WiFi or both), is used to send and receive the traffic. Application layer can make the decision to select a specific transport based on preference e.g. content location, content type, content publisher, congestion, cost, quality of service etc. There can be an Application Programming Interface (API) to exchange parameters required for transport selection. The southbound interactions of TCL will be either to IP or ICN at the network layer. When selecting the IPoICN [TROSSEN] mode, the TCL is responsible for receiving an incoming IP or HTTP packet and publishing the packet under a suitable ICN name (i.e. the hash over the destination IP address for an IP packet or the hash over the FQDN of the HTTP request for an HTTP packet) to the ICN network. In the HTTP case, the TCL maintains a pending request mapping table to map returning responses to the originating HTTP request. The common API will provide a common 'connection' abstraction for this HTTP mode of operation, returning the response over said connection abstraction, akin to the TCP socket interface, while implementing a reliable transport connection semantic over the ICN from the UE to the receiving UE or the PGW. If the HTTP protocol stack remains unchanged, therefore utilizing the TCP protocol for transfer, the TCL operates in local TCP termination mode, retrieving the HTTP packet through said local termination. The southbound interactions of the Transport Convergence Layer (TCL) will be either to IP or ICN at the network layer.

- ICN function (forwarder) is introduced in parallel to the existing IP layer. ICN forwarder contains functional capabilities to forward ICN packets, e.g. Interest packet to gNB or response "data packet" from gNB to the application.

- For dual stack scenario, when UE is not supporting ICN at network layer, we use IP underlay to transport ICN packets. ICN function will use IP interface to send Interest and Data packets for fetching or sending data using ICN protocol function. This interface will use ICN overlay over IP using any overlay tunneling mechanism.

- To support ICN at network layer in UE, PDCP layer has to be aware of ICN capabilities and parameters. PDCP is located in the Radio Protocol Stack in the 5G Air interface, between IP (Network layer) and Radio Link Control Layer (RLC). PDCP performs following functions [TS36.323]:

  * Data transport by listening to upper layer, formatting and pushing down to Radio Link Layer (RLC).

* Header compression and decompression using Robust Header Compression (ROHC).

* Security protections such as ciphering, deciphering and integrity protection.

* Radio layer messages associated with sequencing, packet drop detection and re-transmission etc.

o No changes are required for lower layer such as RLC, MAC and Physical (L1) because they are not IP aware.

5. 5GLAN Architecture with ICN Support

5.1. 5GC Architecture Extensions for 5GLAN Support

In this section, we present an overview of ongoing work to provide cellular LAN connectivity over a 5GC compliant network for Release 16 and above deployments.

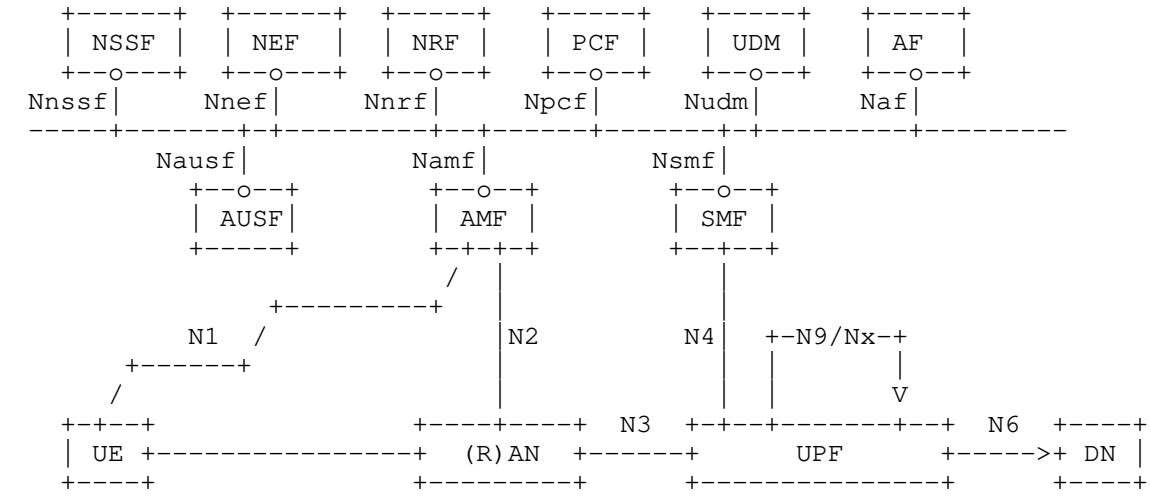


Figure 6: 5G Core Network with Vertical_LAN (5GLAN) Extensions

Figure 6 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 Session Management Function (SMF) 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).

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).

5.1.1. Realization of Nx Interface

In the following, we discuss ongoing work to realize the Nx interface, i.e., path-based forwarding is assumed with 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 bitposition 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 bitposition 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 receiving device, the end source MAC address is restored as the source MAC, creating the perception of a link-local L2 communication between the end source and destination devices.

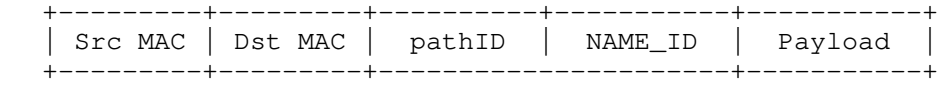


Figure 7: General Packet Structure

For this end-to-end transfer, the general packet structure of Figure 7 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 and the PATH_ID is the bitfield-based path identifier for the path-based forwarding.

5.1.2. Bitfield-based Forwarding in Existing Transport Networks

An emerging technology for Layer 2 forwarding that suits the 5GLAN architecture in Figure 6 is that of Software-defined networking (SDN) [SDNDef], 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 (through the SMF function of the 5GC). In order to forward based on such bitfield path information, the NR instructs the SDN controller 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 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 bitnumber assignment) will require suitable changes of forwarding rules in switches.

Although we focus the methods in this draft on Layer 2 forwarding approaches, 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.

5.2. ICN over 5GLAN

ICN aims at replacing the routing functionality of the Internet Protocol (IP). It is therefore natively supported over a Layer 2 transport network, such as Ethernet-based networks. Deployments exists over WiFi and local LAN networks, while usually overlaying (over IP) is being used for connectivity beyond localized edge networks.

With the emergence of the 5GLAN capability in (future) Release 16 based 5G networks, such cellular LAN connectivity to provide pure ICN could be utilized for pure ICN-based deployments, i.e. without the dual stack capability outlined in Section 4.2.3.2. With this, the entire 5G network would be interpreted as a local LAN, providing the necessary Layer 2 connectivity between the ICN network components. With the support of roaming in 5GLAN, such ’5G network’ can span several operators and therefore large geographies.

Such deployment, however, comes without any core network integration, similar to the one outlined in Section 4.1, and therefore does not utilize ICN capabilities within the overall 5G core and access network. Benefits such as those outlined in the introduction, e.g., caching, would only exist at the endpoint level (from a 5GLAN perspective).

However, ICN components could be provided as SW components in a network slice at the endpoints of such 5GLAN connectivity, utilizing in-network compute facilities, e.g., for caching, CCN routing capabilities and others. Such endpoint-driven realization of a specific ICN deployment scenario is described in more detail in [I-D.trossen-icnrg-IP-over-icn], focusing on the provisioning of IP-based services over an ICN, which in turn is provided over a LAN (and therefore also 5GLAN) based transport network.

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 4 of [I-D.irtf-icnrg-deployment-guidelines]) that is being realized and the 'deployment migration paths' (covered in Section 5 of [I-D.irtf-icnrg-deployment-guidelines]) that are being provided.

The solutions proposed in this draft relate to those 'deployment configuration' as follows:

- The integration with the 5GC, as proposed in Section 4.2, follows the 'Clean-slate ICN' deployment configuration, i.e., integrating the ICN capabilities natively into the 5GC through appropriate extensions at the control and user plane level.

- The utilization of the 5GLAN capabilities, as proposed in Section 5.2, follows the 'ICN-as-an-Overlay', interpreting the 5GLAN as an overlay capability with no 5GC integration being considered (as in the 'Clean-slate ICN' configuration).

- 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 5.2.

In relation of the 'deployment migration paths', the solutions in this draft relate as follows:

- The integration with the 5GC, as proposed in Section 4.2, facilitates 'edge network migration' (interpreting the cellular sub-system here as an edge network albeit a possibly geographically large one).

- The dual-stack deployment, as proposed in Section 4.2.3, facilitates 'application and services migration' through not only supporting ICN applications but also IP-based applications through the proposed IP-over-ICN mapping in the terminal.

- The ICN over 5GLAN deployment, possibly combined with an ICN-as-a-Slice deployment, facilitates the 'content delivery networks migration' through a deployment of 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 explore the feasibility of realizing future networking architectures like ICN within the proposed 3GPP's 5GC architecture. Towards this, we summarized the design principles that offer 5GC the flexibility to enable new network architectures. We then discuss 5GC architecture aspects along with the user/control plane extensions required to handle ICN PDU sessions formally to realize ICN with 5GC integration as well as ICN over a pure 5GLAN connectivity.

8. IANA Considerations

This document requests no IANA actions.

9. Security Considerations

This draft proposes extensions to support ICN in 5G's next generation core architecture. ICN being name based networking opens up new security and privacy considerations which have to be studied in the context of 5GC. This is in addition to other security considerations of 5GC for IP or non-IP based services considered in [TS33.899].

10. Acknowledgments

...

11. Informative References

[H2020] 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.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-06 (work in progress), May 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-03 (work in progress), March 2019.

[I-D.muscariello-intarea-hicn]
Muscariello, L., Carofiglio, G., Auge, J., and M. Papalini, "Hybrid Information-Centric Networking", draft-muscariello-intarea-hicn-01 (work in progress), December 2018.

[I-D.white-icnrg-ipoc]
White, G., Shannigrahi, S., and C. Fan, "Internet Protocol Tunneling over Content Centric Mobile Networks", draft-white-icnrg-ipoc-01 (work in progress), June 2018.

[ICNMOB] Azgin, A., Ravidran, R., Chakraborti, A., and G. Wang, "Seamless Producer Mobility as a Service in Information Centric Networks.", 5G/ICN Workshop, ACM ICN Sigcomm 2016, 2016.

[Jacobson]
Jacobson, V. and et al., "Networking Named Content", Proceedings of ACM Context, , 2009.

[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.

[lteversus5g]
Kim, J., Kim, D., and S. Choi, "3GPP SA2 architecture and functions for 5G mobile communication system.", ICT Express 2017, 2017.

[NFN] Sifalakis, M., Kohler, B., Christopher, C., and C. Tschudin, "An information centric network for computing the distribution of computations", ACM, ICN Sigcomm, 2014.

[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.

[Ravindran]
Ravindran, R., Chakraborti, A., Amin, S., Azgin, A., and G. Wang, "5G-ICN : Delivering ICN Services over 5G using Network Slicing", IEEE Communication Magazine, May, 2016.

[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, Maylaysia, 2016.

[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.

[SDNDef] Open Networking Foundation, available at https://www.opennetworking.org/sdn-definition/, "Software-Defined Networking (SDN) Definition", 2018.

[TROSSEN] Trossen, D., Reed, M., Riihijarvi, J., Georgiades, M., and G. Xylomenos, "IP Over ICN - The Better IP ?", EuCNC, European Conference on Networks and Communications , July, 2015.

[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.

[TS23.799]
3gpp-23.799, "Technical Specification Group Services and System Aspects; Study on Architecture for Next Generation System (Rel. 14)", 3GPP , 2017.

[TS33.899]
3gpp-33.899, "Study on the security aspects of the next generation system", 3GPP , 2017.

[TS36.323]
3gpp-36.323, "Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification (Rel. 15)", 3GPP , January 2019.

[TS38.300]
3gpp-38-300, "Technical Specification Group Radio Access Network; NR; NR and NG-RAN Overall Description; Stage 2 (Rel.15)", 3GPP , January 2019.

[VSER] Ravindran, R., Liu, X., Chakraborti, A., Zhang, X., and G. Wang, "Towards software defined ICN based edge-cloud services", CloudNetworking(CloudNet), IEEE Internation Conference on, IEEE Internation Conference on CloudNetworking(CloudNet), 2013.

Authors' Addresses

Ravi Ravindran
Futurewei Technologies
2330 Central Expressway
Santa Clara 95050
USA

Email: ravi.ravindran@futurewei.com

Prakash Suthar
Cisco Systems
9501 Technology Blvd.
Rosemont 50618
USA

Email: psuthar@cisco.com
URI: http://www.cisco.com/

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/

Greg White
CableLabs
858 Coal Creek Circle
Louisville CO 80027
USA

Email: g.white@cablelabs.com

icnrg
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

L. Wang
L. Geng
China Mobile

July 02, 2018

Consideration on Applying ICN to Edge Computing
draft-wang-icnrg-icn-edge-01

Abstract

Aiming at research on applying Information Centric Networking (ICN) technology to Edge Computing, this document analyzes the reasons and opportunities of applying ICN to EC. As well, towards this end, technical considerations are described and relevant scenarios are shown in the document. Benefits of deploying ICN at edge is analyzed in the document.

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 January 3, 2019.

Table of Contents

1. Introduction

Information Centric Networking (ICN) takes significant technical revolution and fundamental change on communication and networking. It uses content/information centric networking to replace traditional address-centric networking which change the existing networking model essentially. It can also be regarded an Internet structure evolution from host-centric structure to data-centric structure which means accessing data by naming. This structure enables to make the data relating application more independent of its location and transmission method. What's more, security mechanism is based on information instead of host and the caching in forwarding process that promotes huge information transmission efficiently. It is very promising to apply ICN to some popular network architecture.

Meanwhile, Edge Computing (EC) is becoming important network architecture because of its outstanding performance in real-time, reliability, security, etc. It deploys services on the edge of network to be close to consumers, and offers decentralized function to enable excellent properties in local computing, storage, connectivity and so on. At present, Edge Computing works broadly on IoT and industrial verticals such as Energy, Manufacturing, Smart City and Smart Grid.

Therefore, it is worth attempting the possibility of using ICN on EC. ICN naturally supports decentralized caching, self authentication and

multicast that can enable EC deployment. The combination of ICN and EC is able to offer a win-win approach and benefit mutually for maximum performance. In the following sections, we will seek the opportunities of applying ICN to EC, and outline the correlative properties of both. The technical consideration of the approach and relevant scenarios will be described as well.

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 [RFC2119].

EC- Edge Computing, an network architecture that provides local compute, storage and connectivity services

Other ICN related words used in this document are interpreted as description in [ICNRG-Terminology].

3. Opportunitiesof Applying ICN to EC

3.1. ICN Enable Traffic Convergence

In traditional networks most typical service nodes are deployed in centre, so the network flow are transferred from the centre to the edge and downlink traffic is dominant. But as the IoT highly developed, a large amount of devices are deployed on the edge, which, therefore results in considerable uplink traffic. The requirement of traffic service flattening requests the technique that is able to make local communication for traffic convergence. This could be the entry point of applying ICN to EC.

3.2. Functionality Complementary

Both ICN and EC possess some correlative properties, such as decentralized deployment, local communication capacity, producing abundant uplink traffic flow, etc. However, there are also some other properties they posses respectively which are complementary.

For ICN, caching and forwarding are two basic functions which are more about connectivity. But in practical cases, ICN node devices such as gateways demand for light computing and storage functions as well. Light computing and storage can make the network more dynamic, flexible and enable some AI deployments as well. Fortunately, edge computing is able to support storage and computing naturally. A combination of both ICN and edge computing can be mutually benefitted for maximum performance.

3.3. Practicality of ICN Deployment on Edge

TCP/IP network model has been used for quite a while and is worldwide deployed now. No matter according to cost, difficulty, risk or other consideration, it is not realistic to deploy ICN on the whole network. However, the partial deployment of ICN can have a chance, such as ICN over IP or IP over ICN. Deploying ICN on edge service not only can help to mitigate the ICN whole-network deployment complexity, but also makes the network model more flexible.

4. Technical consideration of applying ICN to Edge Computing

4.1. Optimizing EC Network Disconnection Solution

In some scenarios that the network is not able to offer end to end communication such as power failure or natural disasters that could result in the interruption of the network and other local disconnections problems. Often such failure can cause a series of accidents and even chain reaction, resulting in the loss of enterprises and production. In the case, edge computing enable to supply service which is closer to the edge of data generation and business control deployment, making the computation much closer to the data source. Even if the network fails to get connected, the device can rely on local networks for data communication and processing.

However, (a) sometimes the data stored on the edge is "staleness" due to incapacity of updating timely. This could result in the mistake or staleness data transmission. (b) Furthermore, the storage space on edge is limited. For instance, it is not able to update new content if there is no spare space when storage on edge which normally is small storage capacity, is full. This can also cause the (a) problem.

In ICN network, the content is cached along the path it delivery. So the objective content can be from the source node or the other content caching nodes. When the network is disconnected, the caching content or data in decentralized nodes can be used in edge computing. Caching algorithm of ICN is able to solve two problems stated in previous paragraph by updating data efficiently and dynamically. This benefits from the caching replacement policies of ICN. The policies, such as LRU or LFU, provide mechanism how long or how often the data will be updated. Therefore the data is either the newest or the most popular. Decentralized content caching of ICN strengthens EC network disconnection solution and make more flexible networking.

4.2. Reducing Traffic Congestion

In IoT industry, there are a huge number of devices deployed on the edge which result in a significant amount of uplink flow traffic. In EC, the prominent quantity of traffic is easy to cause traffic congestion.

In ICN network, the data content not only from the source node, but also it is cached in other nodes along the delivery path. So when the edge node request the data, it is not necessary to deliver data from the source node. For instance, in the figure, if node 1 is source node. When node3 requests data from node1, the content will be cached in both node A and node B. So next time when node4 needs the same content data, node B will deliver it, and vice versa. In the case, the traffic is not from node1(source node) to node3 or node4 anymore, but mainly from A to B. Therefore, ICN decentralized content caching enable traffic convergence.to reduce traffic congestion.

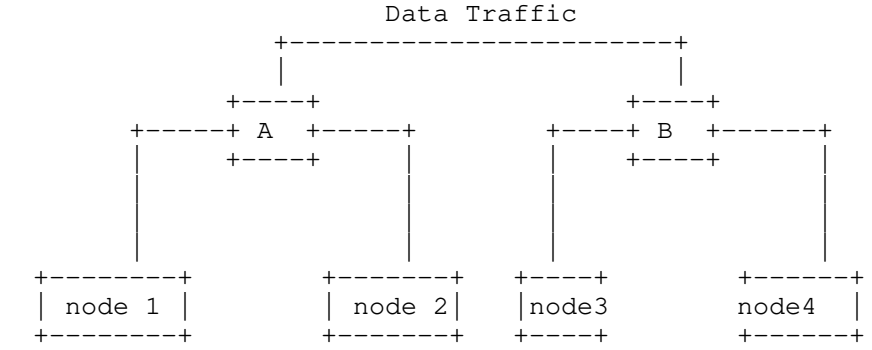


Figure 1

Traffic Convergence in ICN

4.3. Security Consideration of Using ICN

Security problem is crucial and urgent to the EC applications. Firstly, there are many devices on edge are exposed to users which is easy to be attacked. Secondly, although authority level on edge is lower than host and cloud, there are more people can get access to the devices and application. This is in consideration of the consumer convenience and deployment flexibility. Hence, application and services are vulnerable on edge.

Instead of binding security to host node, ICN advocates the model of trust in content. This offers host-independent security mechanism which focuses more on securing information object and content trust. It means host attack no more can interfere edge application. Furthermore, self-certify names model of ICN enable to verify the binding between public key and self-certify name in distributed system without relying on a third party. This can reduce the security risk of involving a third party.

4.4. Content Centric Networking values edge devices

No matter ICN or CCN, they all promote content centric communication model. Independent from host node, naming on edge node gain more valuation on edge devices.

4.5. Partial deployment of ICN on Edge

In consideration of cost and complexity of deploying ICN, it is not necessary to use ICN in the whole network. ICN using on edge is enough to highlight its advantage. Furthermore, there can be a corporation between ICN edge service and IP network.

5. Reverse and Cooperation with CDN

Content Delivery Network (CDN) system, based on IP, composes a couple of servers that deliver content to a user, based on the geographic locations of the user, the origin resource and the CND server nodes. Normally, the resource is distributed in a downlink traffice in figure2.

However, in a ICN network, resource or origin node is not the central node anymore. An edge device can be the origin node that provides the resource which is delivered to ICN servers, and further distributed to the receiver nodes. As a consequence, the routing is from edge to central, or there will be an uplink traffic. This just reverses the CDN Mechanism, which is shown in figure3.

Therefore, deploying ICN network at edge that pulls the requested data from resource edge node to the servers can cooperate with CDN network. An ICN/CDN server is anticipated to translate the protocol between both and deliver the data to the receiver nodes which can be ICN network or IP network.

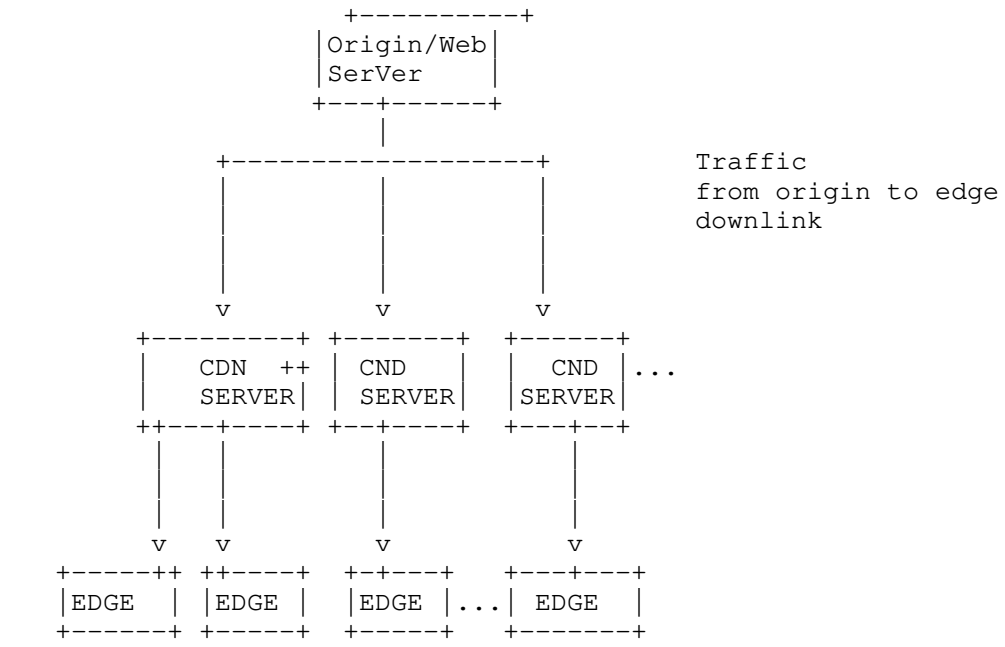


Figure2 CDN Mechanism

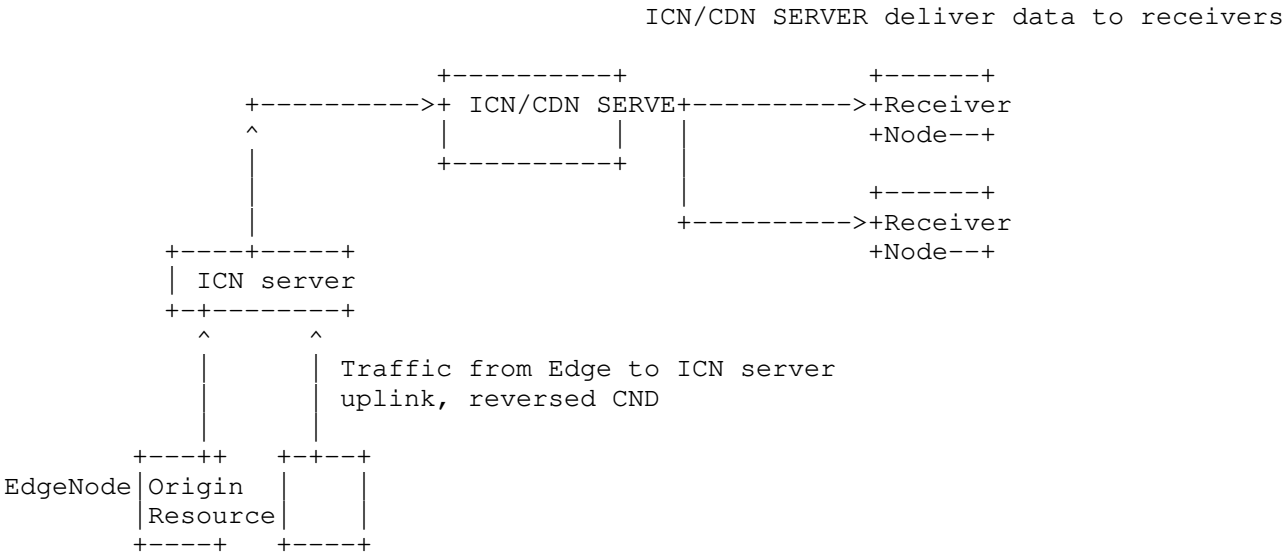


Figure3 ICN Mechanism

6. Conclusion

This draft described the correlative properties of ICN and EC to analyze the opportunity of applying ICN to edge computing. The traffic uplink flow model is the entry point of this research. We could see ICN deployment is beneficial to EC by combining the outstanding performances of both. Furthermore, a win-win model is schemed in the document by means of mutual complementing. However, there are still challenges on deploying ICN on edge such as high speed mobility, fast context resolution and so on. These questions need to be answered in the future.

7. Informative References

[I-D.boucadair-connectivity-provisioning-protocol]
Boucadair, M., Jacquenet, C., Zhang, D., and P. Georgatsos, "Connectivity Provisioning Negotiation Protocol (CPNP)", draft-boucadair-connectivity-provisioning-protocol-15 (work in progress), December 2017.

[ICNRG-Terminology]
"Information-Centric Networking (ICN): CCN and NDN Terminology", <https://datatracker.ietf.org/doc/draft-irtf-icnrg-terminology/>.

[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>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

Authors' Addresses

Lei Wang
China Mobile
Beijing 100053
China

Email: jifengyiwl@163.com

Liang Geng
China Mobile
Beijing 100053
China

Email: gengliang@chinamobile.com

ICNRG
Internet-Draft
Intended status: Experimental
Expires: December 22, 2019

G. White
CableLabs
S. Shannigrahi
C. Fan
Colorado State University

June 20, 2019

Internet Protocol Tunneling over Content Centric Mobile Networks
draft-white-icnrg-ipoc-02

Abstract

This document describes a protocol that enables tunneling of Internet Protocol traffic over a Content Centric Network (CCN) 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 CCN or NDN, this protocol achieves performance on par with LTE-EPC, equivalent efficiency, and substantially lower implementation and protocol complexity. Furthermore, the use of CCN/NDN for this purpose paves the way for the deployment of ICN native applications on the mobile network.

Table of Contents

1. Introduction

Content Centric Networking 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, CCN elegantly supports information retrieval by mobile client devices without the need for tunneling or a location registration protocol. Furthermore, CCN supports a client device utilizing multiple network attachments (e.g. multiple radio links) simultaneously in order to provide greater reliability or greater performance. Finally, CCN is optimized for content retrieval, where content can be easily retrieved from an on-path cache.

A significant hurdle that stands in the way of deploying a CCN-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 CCN 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 CCN. 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 CCN 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 (CCN) that natively supports mobility, application developers have the option to update their applications to run directly over CCN, gaining all of the advantages that come with this new protocol.

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. CCN/NDN Overview

In the CCN (or NDN) 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 CCN 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 CCN 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 CCN 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 CCN 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 CCN Interest messages, and then sends them into the CCN 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 CCN Content Object messages, and then launched into the CCN 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 CCN, and as a result uses the optional interest payload mechanism described in the "Consumer Behavior" section of [I-D.irtf-icnrg-ccnxsemantics]. 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 CCN 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 CCN 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:

- ccnx:/ipoc - The name prefix used in all IPoC messages
- zone_id - An optional zone identifier to allow for zone-based IP address re-use.
- 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 CCN 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.
- 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:

- Initialize IDC to a startup value: INIT_IDC.

- 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:

- An Interest is received with the /init/<random_string> name.
- 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.
- 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

TBD.

15. IANA Considerations

This document has no actions for IANA.

16. Normative References

[I-D.irtf-icnrg-ccnxmessages]
Mosko, M., Solis, I., and C. Wood, "CCNx Messages in TLV Format", draft-irtf-icnrg-ccnxmessages-09 (work in progress), January 2019.

[I-D.irtf-icnrg-ccnxsemantics]
Mosko, M., Solis, I., and C. Wood, "CCNx Semantics", draft-irtf-icnrg-ccnxsemantics-10 (work in progress), January 2019.

[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>.

[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>.

Authors' Addresses

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

Email: g.white@cablelabs.com

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

Email: susmit@colostate.edu

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

Email: chengyu.fan@colostate.edu