Stream: Internet Research Task Force (IRTF)
RFC: 9139
Category: Experimental
Published: November 2021
ISSN: 2070-1721
Authors: C. Gündoğan *HAW Hamburg*; T. Schmidt *HAW Hamburg*; M. Wählisch *link-lab & FU Berlin*; C. Scherb *FHNW*
C. Marxer *University of Basel*; C. Tschudin *University of Basel*

RFC 9139
Information-Centric Networking (ICN) Adaptation to Low-Power Wireless Personal Area Networks (LoWPANs)

Abstract

This document defines a convergence layer for Content-Centric Networking (CCNx) and Named Data Networking (NDN) over IEEE 802.15.4 Low-Power Wireless Personal Area Networks (LoWPANs). 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 IPv6 over LoWPAN (6LoWPAN) is extended to include new dispatch types for CCNx and NDN. Additionally, the fragmentation component of the 6LoWPAN dispatching framework is applied to Information-Centric Network (ICN) chunks. In its second part, the document defines stateless and stateful compression schemes to improve efficiency on constrained links. Stateless compression reduces TLV expressions to static header fields for common use cases. Stateful compression schemes elide states local to the LoWPAN and replace names in Data packets by short local identifiers.

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

The IEEE 802.15.4 [ieee802.15.4] link layer is used in low-power and lossy networks (see `LLN` in [RFC7228]), in which devices are typically battery operated and constrained in resources. Characteristics of LLNs include an unreliable environment, low-bandwidth transmissions, and increased latencies. IEEE 802.15.4 admits a maximum physical-layer packet size of 127 bytes. The maximum frame header size is 25 bytes, which leaves 102 bytes for the payload. IEEE 802.15.4 security features further reduce this payload length by up to 21 bytes, yielding a net of 81 bytes for CCNx or NDN packet headers, signatures, and content.

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

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

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

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

```
                          |Gateway Services|
                          -------------------------
                                |
                            ,--------,
                            |        |
                            |   BR   |
                            |        |
                            '--------'
                                         LoWPAN
                          O           O
                                O
                        O                O   embedded
                          O     O     O      devices
                           O         O
```

Figure 1: IoT Stub Network

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

2. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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
NDN:	Named Data Networking
byte:	synonym for octet
nibble:	synonym for 4 bits
time-value:	a time offset measured in seconds
time-code:	an 8-bit encoded time-value

3. Overview of ICN LoWPAN

3.1. Link-Layer Convergence

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

```
      Device 1                                         Device 2
                               Router
,------------------,                             ,------------------,
|  Application   . |     __________________      | ,-> Application  |
|----------------|-|    |    NDN / CCNx    |     |-|----------------|
|  NDN / CCNx    | |    | ,--------------, |     | |    NDN / CCNx  |
|----------------|-|    |-|--------------|-|     |-|----------------|
|  ICN LoWPAN    | |    | |  ICN LoWPAN  | |     | |    ICN LoWPAN  |
|----------------|-|    |-|--------------|-|     |-|----------------|
|  Link Layer    | |    | |  Link Layer  | |     | |    Link Layer  |
'----------------|-'    '-|--------------|-'     '-|----------------'
                 '--------'              '---------'
```

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 preconfiguration of a global state.

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

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

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

```
Uncompressed:

   Variable-length TLV      Fixed-length TLV      Boolean TLV
,-----------------------,-----------------------,-------------,
+-------+-------+-------+-------+-------+-------+------+------+
|  TYP  |  LEN  |  VAL  |  TYP  |  LEN  |  VAL  |  TYP | LEN  |
+-------+-------+-------+-------+-------+-------+------+------+

Compressed:

  +---+---+---+---+---+---+---+---+
  | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |  Bit Field
  +---+---+---+---+---+---+---+---+
    |       |                   |
 ,--'       '----,              '- Boolean Value
 |               |
+-------+-------+-------+
|  LEN  |  VAL  |  VAL  |
+-------+-------+-------+
'---------------'-------'
  Var-len Value  Fixed-len Value
```

Figure 3: Compression Using a Compact Bit Field -- Bits Encode the Inclusion of TLVs

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

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

3.3. Stateful Header Compression

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

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

4. IEEE 802.15.4 Adaptation

4.1. LoWPAN Encapsulation

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

[RFC8025] further specifies dispatch Pages to switch between different contexts. When a LoWPAN parser encounters a `Page switch` LoWPAN encapsulation header, all following encapsulation headers are interpreted by using a dispatch Page, as specified by the `Page switch` header. Pages 0 and 1 are reserved for 6LoWPAN. This document uses Page 14 (`1111 1110 (0xFE)`) for ICN LoWPAN.

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

```
  0   1   2   3   ...
+---+---+---+---+---
| 0 | P | M | C |
+---+---+---+---+---
```

Figure 4: Base Dispatch Format for ICN LoWPAN

P: Protocol
- 0: The included protocol is NDN.
- 1: The included protocol is CCNx.

M: Message Type
- 0: The payload contains an Interest message.
- 1: The payload contains a Data message.

C: Compression
- 0: The message is uncompressed.
- 1: The message is compressed.

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

```
  0   1   2   3       ... ...
+---+---+---+---+...+---+---+
| 0 | P | M | 1 |   |CID|EXT|
+---+---+---+---+...+---+---+
```

Figure 5: Extended Dispatch Format for Compressed ICN LoWPAN

CID: Context Identifier
- 0: No context identifiers are present.
- 1: Context identifier(s) are present (see Section 8.1).

EXT: Extension
- 0: No extension bytes are present.
- 1: Extension byte(s) are present (see Section 4.1.1).

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

```
    +-------...------+------...-----+--------+-------...-------+-----...
    | IEEE 802.15.4 | RFC4944 Disp.|  Page   | ICN LoWPAN Disp.| Payl. /
    +-------...------+------...-----+--------+-------...-------+-----...
```

Figure 6: LoWPAN Encapsulation with ICN LoWPAN

IEEE 802.15.4:	The IEEE 802.15.4 header.
RFC4944 Disp.:	Optional additional dispatches defined in Section 5.1 of [RFC4944].
Page:	`Page switch`. 14 for ICN LoWPAN.
ICN LoWPAN:	Dispatches as defined in Sections 5 and 6.
Payload:	The actual (un-)compressed CCNx or NDN message.

4.1.1. Dispatch Extensions

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

```
                  0   1   2   3   4   5   6   7
                +---+---+---+---+---+---+---+---+
                | - | - | - | - | - | - | - |EXT|
                +---+---+---+---+---+---+---+---+
```

Figure 7: Base Format for Dispatch Extensions

EXT: Extension

0:	No other extension byte follows.
1:	A further extension byte follows.

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

4.2. Adaptation-Layer Fragmentation

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

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

```
   +------...------+----...----+--------+------...-------+--------...
   | IEEE 802.15.4 | Frag. 1st |  Page  |   ICN LoWPAN   | Payload  /
   +------...------+----...----+--------+------...-------+--------...

   +------...------+----...----+--------...
   | IEEE 802.15.4 | Frag. 2nd | Payload  /
   +------...------+----...----+--------...

                   .
                   .
                   .

   +------...------+----...----+--------...
   | IEEE 802.15.4 | Frag. Nth | Payload  /
   +------...------+----...----+--------...
```

Figure 8: Fragmentation Scheme

```
         +------...------+--------+------...-------+--------...
         | IEEE 802.15.4 |  Page  |   ICN LoWPAN   | Payload  /
         +------...------+--------+------...-------+--------...
```

Figure 9: Reassembled ICN LoWPAN Frame

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

5. Space-Efficient Message Encoding for NDN

5.1. TLV Encoding

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

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

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

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

Table 1: NDN TLV Encoding Compared to SDNVs

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

5.2. Name TLV Compression

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

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

        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |0 0 1 1|0 1 0 0|       H       |       A       |       W       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |       R       |       o       |       o       |       m       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |0 0 1 1|0 1 0 1|       4       |       8       |       1       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |       H       |       u       |       m       |       i       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |       d       |0 0 1 0|0 0 0 0|       9       |       9       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

5.3. Interest Messages

5.3.1. Uncompressed Interest Messages

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

```
                       0   1   2   3   4   5   6   7
                     +---+---+---+---+---+---+---+---+
                     | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
                     +---+---+---+---+---+---+---+---+
```

Figure 11: Dispatch Format for Uncompressed NDN Interest Messages

5.3.2. Compressed Interest Messages

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

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

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

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

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

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

        +---------+---------+                     +---------+
        |  Msg T  |  Msg L  |                     |  Msg Lc |
        +---------+---------+---------+           +---------+
        | Name T  | Name L  | Name V  |           | Name Vc |
        +---------+---------+---------+           +---------+---------+
        : CBPfx T : CBPfx L :                     : FWDH Lc : FWDH Vc :
        +---------+---------+                     +---------+---------+
        : MBFr T  : MBFr L  :                     |  HPL V  |
        +---------+---------+---------+   ==>     +---------+---------+
        : FWDH T  : FWDH L  : FWDH V  :           :  APM Lc : APM Vc  :
        +---------+---------+---------+           +---------+---------+
        : NONCE T : NONCE L : NONCE V :           : NONCE V :
        +---------+---------+---------+           +---------+
        :  ILT T  :  ILT L  :  ILT V  :           :  ILT Vc :
        +---------+---------+---------+           +---------+
        :  HPL T  :  HPL L  :  HPL V  :
        +---------+---------+---------+
        :  APM T  :  APM L  :  APM V  :
        +---------+---------+---------+
```

Figure 12: Compression of NDN LoWPAN Interest Message

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

```
     0                                       1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | 0 | 0 | 0 | 1 |PFX|FRE|FWD|APM|DIG|        RSV        |CID|EXT|
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 13: Dispatch Format for Compressed NDN Interest Messages

PFX: CanBePrefix TLV
- 0: The uncompressed message does not include a CanBePrefix TLV.
- 1: The uncompressed message does include a CanBePrefix TLV and is removed from the compressed message.

FRE: MustBeFresh TLV
- 0: The uncompressed message does not include a MustBeFresh TLV.
- 1: The uncompressed message does include a MustBeFresh TLV and is removed from the compressed message.

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

APM: ApplicationParameters TLV
- 0: The uncompressed message does not include an ApplicationParameters TLV.
- 1: The uncompressed message does include an ApplicationParameters TLV. The Type field is removed from the compressed message.

DIG: ImplicitSha256DigestComponent TLV
- 0: The name does not include an ImplicitSha256DigestComponent as the last TLV.
- 1: The name does include an ImplicitSha256DigestComponent as the last TLV. The Type and Length fields are omitted.

RSV: Reserved
  Must be set to 0.

CID: Context Identifier
  See Figure 5.

EXT: Extension
- 0: No extension byte follows.
- 1: Extension byte `EXT_0` follows immediately. See Section 5.3.3.

5.3.3. Dispatch Extension

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |         RSV       |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 14: EXT_0 Format

NCS: Name Compression Strategy

- 00: Names are compressed with the default name compression strategy (see Section 5.2).
- 01: Reserved.
- 10: Reserved.
- 11: Reserved.

RSV: Reserved

Must be set to 0.

EXT: Extension

- 0: No extension byte follows.
- 1: A further extension byte follows immediately.

5.4. Data Messages

5.4.1. Uncompressed Data Messages

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
```

Figure 15: Dispatch Format for Uncompressed NDN Data Messages

5.4.2. Compressed Data Messages

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

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

1. The `Type` field of the outermost MessageType TLV is removed.
2. The Name TLV is compressed according to Section 5.2. For this, all NameComponents are expected to be of type GenericNameComponent and to have a length greater than 0. In any other case, the message **MUST** be sent uncompressed.
3. The MetaInfo TLV Type and Length fields are elided from the compressed Data message.
4. The FreshnessPeriod TLV **MUST** be moved to the end of the compressed Data message. Type and Length fields are elided, and the value is encoded as described in Section 7 as a 1-byte time-code. If the freshness period is not a valid time-value, then the message **MUST** be sent uncompressed in order to preserve the security envelope of the Data message. The presence of a FreshnessPeriod TLV is deduced from the remaining one-byte length to parse.
5. The Type fields of the SignatureInfo TLV, SignatureType TLV, and SignatureValue TLV are removed.

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

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

         +---------+---------+                 +---------+
         |  Msg T  |  Msg L  |                 |  Msg Lc |
         +---------+---------+---------+       +---------+
         | Name T  | Name L  | Name V  |       | Name Vc |
         +---------+---------+---------+       +---------+---------+
         : Meta T  : Meta L  :                 : CTyp Lc : CTyp V  :
         +---------+---------+---------+       +---------+---------+
         : CTyp T  : CTyp L  : CTyp V  :       : FBID V  :
         +---------+---------+---------+  ==>  +---------+---------+
         : FrPr T  : FrPr L  : FrPr V  :       : CONT Lc : CONT V  :
         +---------+---------+---------+       +---------+---------+
         : FBID T  : FBID L  : FBID V  :       |  Sig Lc |
         +---------+---------+---------+       +---------+---------+
         : CONT T  : CONT L  : CONT V  :       | SInf Lc | SInf Vc |
         +---------+---------+---------+       +---------+---------+
         |  Sig T  |  Sig L  |                 | SVal Lc | SVal Vc |
         +---------+---------+---------+       +---------+---------+
         | SInf T  | SInf L  | SInf V  |       : FrPr Vc :
         +---------+---------+---------+       +---------+
         | SVal T  | SVal L  | SVal V  |
         +---------+---------+---------+
```

Figure 16: Compression of NDN LoWPAN Data Message

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

```
     0                                       1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | 0 | 0 | 1 | 1 |FBI|CON|KLO|            RSV            |CID|EXT|
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 17: Dispatch Format for Compressed NDN Data Messages

FBI: FinalBlockId TLV

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

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

CON: ContentType TLV

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

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

KLO: KeyLocator TLV

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

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

RSV: Reserved

Must be set to 0.

CID: Context Identifier

See Figure 5.

EXT: Extension

0: No extension byte follows.

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

5.4.3. Dispatch Extension

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

```
                     0   1   2   3   4   5   6   7
                   +---+---+---+---+---+---+---+---+
                   |  NCS  |          RSV      |EXT|
                   +---+---+---+---+---+---+---+---+
```

Figure 18: EXT_0 Format

NCS: Name Compression Strategy

00:	Names are compressed with the default name compression strategy (see Section 5.2).
01:	Reserved.
10:	Reserved.
11:	Reserved.

RSV: Reserved
Must be set to 0.

EXT: Extension

0:	No extension byte follows.
1:	A further extension byte follows immediately.

6. Space-Efficient Message Encoding for CCNx

6.1. TLV Encoding

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

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

6.2. Name TLV Compression

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

6.3. Interest Messages

6.3.1. Uncompressed Interest Messages

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
```

Figure 19: Dispatch Format for Uncompressed CCNx Interest Messages

6.3.2. Compressed Interest Messages

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

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

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

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

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

    +-----------------------------+          +-------------------------+
    |    Uncompr. Fixed Header    |          |   Compr. Fixed Header   |
    +-----------------------------+          +-------------------------+
    +---------+---------+---------+          +---------+
    : ILT T   : ILT L   : ILT V   :          : ILT Vc  :
    +---------+---------+---------+          +---------+
    : MSGH T  : MSGH L  : MSGH V  :          : MSGH Vc :
    +---------+---------+---------+          +---------+
    +---------+---------+                    +---------+
    | MSGT T  | MSGT L  |                    | Name Vc |
    +---------+---------+---------+          +---------+
    | Name T  | Name L  | Name V  |    ==>   : KIDR Vc :
    +---------+---------+---------+          +---------+
    : KIDR T  : KIDR L  : KIDR V  :          : OBHR Vc :
    +---------+---------+---------+          +---------+---------+
    : OBHR T  : OBHR L  : OBHR V  :          : PAYL Lc : PAYL V  :
    +---------+---------+---------+          +---------+---------+
    : PAYL T  : PAYL L  : PAYL V  :          : VALG Lc : VALG Vc :
    +---------+---------+---------+          +---------+---------+
    : VALG T  : VALG L  : VALG V  :          : VPAY Lc : VPAY V  :
    +---------+---------+---------+          +---------+---------+
    : VPAY T  : VPAY L  : VPAY V  :
    +---------+---------+---------+
```

Figure 20: Compression of CCNx LoWPAN Interest Message

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

```
     0                                       1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | 0 | 1 | 0 | 1 |FLG|PTY|HPL|FRS|PAY|ILT|MGH|KIR|CHR|VAL|CID|EXT|
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 21: Dispatch Format for Compressed CCNx Interest Messages

FLG: Flags field in the fixed header

0: The Flags field equals 0 and is removed from the Interest message.

1: The Flags field appears in the fixed header.

PTY: PacketType field in the fixed header

0: The PacketType field is elided and assumed to be `PT_INTEREST`.

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

HPL: HopLimit field in the fixed header

0: The HopLimit field appears in the fixed header.

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

FRS: Reserved field in the fixed header

0: The Reserved field appears in the fixed header.

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

PAY: Optional Payload TLV

0: The Payload TLV is absent.

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

ILT: Optional hop-by-hop InterestLifetime TLV

See Section 6.3.2.1 for further details on the ordering of hop-by-hop TLVs.

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

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

MGH: Optional hop-by-hop MessageHash TLV

See Section 6.3.2.1 for further details on the ordering of hop-by-hop TLVs.

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

0: The MessageHash TLV is absent.

1: A T_SHA-256 TLV is present, and the Type and Length fields are removed. The Length field is assumed to represent 32 bytes. The outer Message Hash TLV is omitted.

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

0: The KeyIdRestriction TLV is absent.

1: A T_SHA-256 TLV is present, and the Type and Length fields are removed. The Length field is assumed to represent 32 bytes. The outer KeyIdRestriction TLV is omitted.

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

0: The ContentObjectHashRestriction TLV is absent.

1: A T_SHA-256 TLV is present, and the Type and Length fields are removed. The Length field is assumed to represent 32 bytes. The outer ContentObjectHashRestriction TLV is omitted.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs

0: No validation-related TLVs are present in the Interest message.

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

CID: Context Identifier
See Figure 5.

EXT: Extension

0: No extension byte follows.

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

6.3.2.1. Hop-By-Hop Header TLVs Compression

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

1. InterestLifetime TLV
2. Message Hash TLV

Note: All hop-by-hop header TLVs other than the InterestLifetime and MessageHash TLVs remain uncompressed in the encoded message, and they appear after the InterestLifetime and MessageHash TLVs in the same order as in the original message.

6.3.2.2. Validation

```
 0       1       2       3       4       5       6       7       8
 +-------+-------+-------+-------+-------+-------+-------+-------+
 |         ValidationAlg         |     KeyID     |      RSV      |
 +-------+-------+-------+-------+-------+-------+-------+-------+
```

Figure 22: Dispatch for Interest Validations

ValidationAlg: Optional ValidationAlgorithm TLV

0000: An uncompressed ValidationAlgorithm TLV is included.

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

0010: A T_CRC32C ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included. Additionally, a SignatureTime TLV is inlined without a Type and a Length field.

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

0100: A T_HMAC-SHA256 ValidationAlgorithm TLV is assumed, but no ValidationAlgorithm TLV is included. Additionally, a SignatureTime 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 and Length fields are removed. The Length field is assumed to represent 32 bytes. The outer KeyID TLV is omitted.

11: A T_SHA-512 TLV is present, and the Type and Length fields are removed. The Length field is assumed to represent 64 bytes. The outer KeyID TLV is omitted.

RSV: Reserved
Must be set to 0.

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

6.3.3. Dispatch Extension

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |         RSV       |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 23: EXT_0 Format

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

RSV: Reserved
Must be set to 0.

EXT: Extension
- 0: No extension byte follows.
- 1: A further extension byte follows immediately.

6.4. Content Objects

6.4.1. Uncompressed Content Objects

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
```

Figure 24: Dispatch Format for Uncompressed CCNx Content Objects

6.4.2. Compressed Content Objects

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

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

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

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

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

+------------------------------+           +--------------------------+
|    Uncompr. Fixed Header     |           |   Compr. Fixed Header    |
+------------------------------+           +--------------------------+
+---------+---------+---------+            +---------+
: RCT T   : RCT L   : RCT V   :            : RCT Vc  :
+---------+---------+------.--+            +---------+
: MSGH T  : MSGH L  : MSGH V  :            : MSGH Vc :
+---------+---------+---------+            +---------+
+---------+---------+                      +---------+
| MSGT T  | MSGT L  |                      | Name Vc |
+---------+---------+---------+            +---------+
| Name T  | Name L  | Name V  |    ==>     : EXPT Vc :
+---------+---------+---------+            +---------+---------+
: PTYP T  : PTYP L  : PTYP V  :            : PAYL Lc : PAYL V  :
+---------+---------+---------+            +---------+---------+
: EXPT T  : EXPT L  : EXPT V  :            : VALG Lc : VALG Vc :
+---------+---------+---------+            +---------+---------+
: PAYL T  : PAYL L  : PAYL V  :            : VPAY Lc : VPAY V  :
+---------+---------+---------+            +---------+---------+
: VALG T  : VALG L  : VALG V  :
+---------+---------+---------+
: VPAY T  : VPAY L  : VPAY V  :
+---------+---------+---------+
```

Figure 25: Compression of CCNx LoWPAN Data Message

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

```
    0                                       1
    0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  | 0 | 1 | 1 | 1 |FLG|FRS|PAY|RCT|MGH| PLTYP |EXP|VAL|RSV|CID|EXT|
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 26: Dispatch Format for Compressed CCNx Content Objects

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

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

PAY: Optional Payload TLV
: See Section 6.3.2.

RCT: Optional hop-by-hop Recommended Cache Time TLV

- 0: The Recommended Cache Time TLV is absent.
- 1: The Recommended Cache Time TLV is present, and the Type and Length fields are elided.

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

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

- 0: The MessageHash TLV is absent.
- 1: A T_SHA-256 TLV is present, and the Type and Length fields are removed. The Length field is assumed to represent 32 bytes. The outer Message Hash TLV is omitted.

PLTYP: Optional PayloadType TLV

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

EXP: Optional ExpiryTime TLV

- 0: The ExpiryTime TLV is absent.
- 1: The ExpiryTime TLV is present, and the Type and Length fields are elided.

VAL: Optional ValidationAlgorithm and ValidationPayload TLVs
: See Section 6.3.2.

RSV: Reserved
: Must be set to 0.

CID: Context Identifier
See Figure 5.

EXT: Extension

0: No extension byte follows.

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

6.4.2.1. Hop-By-Hop Header TLVs Compression

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

1. Recommended Cache Time TLV
2. Message Hash TLV

Note: All hop-by-hop header TLVs other than the RecommendedCacheTime and MessageHash TLVs remain uncompressed in the encoded message, and they appear after the RecommendedCacheTime and MessageHash TLVs in the same order as in the original message.

6.4.3. Dispatch Extension

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|  NCS  |         RSV       |EXT|
+---+---+---+---+---+---+---+---+
```

Figure 27: EXT_0 Format

NCS: Name Compression Strategy

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

01: Reserved.

10: Reserved.

11: Reserved.

RSV: Reserved
Must be set to 0.

EXT: Extension

0: No extension byte follows.

1: A further extension byte follows immediately.

7. Compressed Time Encoding

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

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

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

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

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

This configuration allows for the following ranges:

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

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

8. Stateful Header Compression

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

8.1. LoWPAN-Local State

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| X |             CID           |
+---+---+---+---+---+---+---+---+
```

Figure 28: Context Identifier

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

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

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

8.2. En Route State

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

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| X |          HopID            |
+---+---+---+---+---+---+---+---+
```

Figure 29: Context Identifier as HopID

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

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

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

```
    PIT of B      PIT Extension          PIT of C      PIT Extension
+--------+------++------+------+     +--------+------++------+------+
| Prefix | Face || HIDi | HIDo |     | Prefix | Face || HIDi | HIDo |
+========+======++======+======+     +========+======++======+======+
|  /p0   | F_A  || h_A  | h_B  |     |  /p0   | F_A  || h_A  |      |
+--------+------++------+------+     +--------+------++------+------+
                    ^       |                            ^
              store |       '----------------------, ,---' store
                    |                 send         v |
,---,         /p0, h_A          ,---,         /p0, h_B          ,---,
| A | ------------------------> | B | ------------------------> | C |
'---'                           '---'                           '---'
```

Figure 30: Setting Compression State En Route (Interest)

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

```
    PIT of B      PIT Extension          PIT of C      PIT Extension
+--------+------++------+------+     +--------+------++------+------+
| Prefix | Face || HIDi | HIDo |     | Prefix | Face || HIDi | HIDo |
+========+======++======+======+     +========+======++======+======+
|  /p0   | F_A  || h_A  | h_B  |     |  /p0   | F_A  || h_A  |      |
+--------+------++------+------+     +--------+------++------+------+
                    |       ^                            |
               send |       '----------------------, ,---' send
                    v                 match        | v
,---,              h_A          ,---,              h_B          ,---,
| A | <------------------------ | B | <------------------------ | C |
'---'                           '---'                           '---'
```

Figure 31: Eliding Name TLVs Using En Route State (Data)

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

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

8.3. Integrating Stateful Header Compression

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

```
...-------+--------+-------...-------+--...-+-------...
/  ...    |  Page  | ICN LoWPAN Disp.| CIDs | Payload /
...-------+--------+-------...-------+--...-+-------...
```

Figure 32: LoWPAN Encapsulation with ICN LoWPAN and CIDs

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

The HopID is always included as the very first CID.

```
+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+
|1| CID / HopID | --> |1|     CID     | --> |0|     CID     |
+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+
```

Figure 33: Chaining of Context Identifiers

9. ICN LoWPAN Constants and Variables

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

DEFAULT_NDN_HOPLIMIT:	255

10. Implementation Report and Guidance

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

10.1. Preferred Configuration

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

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

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

10.2. Further Experimental Deployments

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

- The name compression scheme (Section 5.2) is optimized for short name components of type GenericNameComponent. An empirical study on name lengths in different deployments of selected use cases, such as smart home, smart city, and industrial IoT can provide meaningful reports on necessary name component types and lengths. A conclusive outcome helps to understand whether and how extension mechanisms are needed (Section 5.3.3). As a preliminary analysis, [ICNLOWPAN] investigates the effectiveness of the proposed compression scheme with URLs obtained from the WWW. Studies on deployments of Constrained Application Protocol (CoAP) [RFC7252] can offer additional insights on naming schemes in the IoT.
- The fragmentation scheme (Section 4.2) inherited from 6LoWPAN allows for a transparent, hop-wise reassembly of CCNx or NDN packets. Fragment forwarding [RFC8930] with selective fragment recovery [RFC8931] can improve the end-to-end latency and reliability while it reduces buffer requirements on forwarders. Initial evaluations [SFR-ICNLOWPAN] show that a naive integration of these upcoming fragmentation features into ICN LoWPAN renders the hop-wise content replication inoperative, since Interest and Data messages are reassembled end-to-end. More deployment experiences are necessary to gauge the feasibility of different fragmentation schemes in ICN LoWPAN.
- The context state (Section 8.1) holds information that is shared between a set of devices in a LoWPAN. Fixed name prefixes and suffixes are good candidates to be distributed to all nodes in order to elide them from request and response messages. More experience and a deeper inspection of currently available and upcoming protocol features is necessary to identify other protocol fields.
- The distribution and synchronization of the context state can potentially be adopted from Section 7.2 of [RFC6775] but requires further evaluations. While 6LoWPAN uses the Neighbor Discovery protocol to disseminate state, CCNx and NDN deployments are missing out on a standard mechanism to bootstrap and manage configurations.
- The stateful en route compression (Section 8.2) supports a limited number of 127 distinct HopIDs that can be simultaneously in use on a single node. Complex deployment scenarios that make use of multiple, concurrent requests can provide a better insight on the number of open requests stored in the PIT of memory-constrained devices. This number can serve as an upper bound and determines whether the HopID length needs to be resized to fit more HopIDs at the cost of additional header overhead.
- Multiple implementations that generate and deploy the compression options of this memo in different ways will also add to the experience and understanding of the benefits and limitations of the proposed schemes. Different reports can help to illuminate the complexity of implementing ICN LoWPAN for constrained devices, as well as on maintaining interoperability with other implementations.

11. Security Considerations

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

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

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

12. IANA Considerations

12.1. Updates to the 6LoWPAN Dispatch Type Field Registry

IANA has assigned dispatch values for ICN LoWPAN in the "Dispatch Type Field" subregistry [RFC4944] [RFC8025] of the "IPv6 Low Power Personal Area Network Parameters" registry. Table 2 represents the updates to the registry.

Bit Pattern	Page	Header Type	Reference
00 000000	14	Uncompressed NDN Interest messages	RFC 9139
00 01xxxx	14	Compressed NDN Interest messages	RFC 9139
00 100000	14	Uncompressed NDN Data messages	RFC 9139
00 11xxxx	14	Compressed NDN Data messages	RFC 9139
01 000000	14	Uncompressed CCNx Interest messages	RFC 9139
01 01xxxx	14	Compressed CCNx Interest messages	RFC 9139
01 100000	14	Uncompressed CCNx Content Object messages	RFC 9139
01 11xxxx	14	Compressed CCNx Content Object messages	RFC 9139

Table 2: Dispatch Types for NDN and CCNx

13. References

13.1. Normative References

[IEEE.754.2019] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2019, <https://standards.ieee.org/content/ieee-standards/en/standard/754-2019.html>.

[ieee802.15.4] IEEE, "IEEE Standard for Low-Rate Wireless Networks", IEEE Std 802.15.4-2020, <https://standards.ieee.org/standard/802_15_4-2020.html>.

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

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, DOI 10.17487/RFC5497, March 2009, <https://www.rfc-editor.org/info/rfc5497>.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, DOI 10.17487/RFC6256, May 2011, <https://www.rfc-editor.org/info/rfc6256>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <https://www.rfc-editor.org/info/rfc6775>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[CCN-LITE] "CCN-lite, a lightweight implementation of the CCNx protocol and its variations", <https://github.com/cn-uofbasel/ccn-lite>.

[ICNLOWPAN] Gündoğan, C., Kietzmann, P., Schmidt, T., and M. Wählisch, "Designing a LoWPAN convergence layer for the Information Centric Internet of Things", Computer Communications, Vol. 164, No. 1, p. 114–123, Elsevier, December 2020, <https://doi.org/10.1016/j.comcom.2020.10.002>.

[ICNRG-FLIC] Tschudin, C., Wood, C., Mosko, M., and D. Oran, Ed., "File-Like ICN Collections (FLIC)", Work in Progress, Internet-Draft, draft-irtf-icnrg-flic-02, 4 November 2019, <https://datatracker.ietf.org/doc/html/draft-irtf-icnrg-flic-02>.

[NDN] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking named content", 5th Int. Conf. on emerging Networking Experiments and Technologies (ACM CoNEXT), December 2009, <https://doi.org/10.1145/1658939.1658941>.

[NDN-EXP1] Baccelli, E., Mehlis, C., Hahm, O., Schmidt, TC., and M. Wählisch, "Information centric networking in the IoT: experiments with NDN in the wild", Proc. of 1st ACM Conf. on Information-Centric Networking (ICN-2014) ACM DL, pp. 77-86, September 2014, <http://dx.doi.org/10.1145/2660129.2660144>.

[NDN-EXP2] Gündoğan, C., Kietzmann, P., Lenders, M., Petersen, H., Schmidt, TC., and M. Wählisch, "NDN, CoAP, and MQTT: a comparative measurement study in the IoT", Proc. of 5th ACM Conf. on Information-Centric Networking (ICN-2018) ACM DL, pp. 159-171, September 2018, <https://doi.org/10.1145/3267955.3267967>.

[NDN-MAC] Kietzmann, P., Gündoğan, C., Schmidt, TC., Hahm, O., and M. Wählisch, "The need for a name to MAC address mapping in NDN: towards quantifying the resource gain", Proc. of 4th ACM Conf. on Information-Centric Networking (ICN-2017) ACM DL, pp. 36-42, September 2017, <https://doi.org/10.1145/3125719.3125737>.

[NDN-PACKET-SPEC] "NDN Packet Format Specification", <https://named-data.net/doc/NDN-packet-spec/0.3/>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7476] Pentikousis, K., Ed., Ohlman, B., Corujo, D., Boggia, G., Tyson, G., Davies, E., Molinaro, A., and S. Eum, "Information-Centric Networking: Baseline Scenarios", RFC 7476, DOI 10.17487/RFC7476, March 2015, <https://www.rfc-editor.org/info/rfc7476>.

[RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. 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>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

[RFC8930] Watteyne, T., Ed., Thubert, P., Ed., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multi-Hop IPv6 Network", RFC 8930, DOI 10.17487/RFC8930, November 2020, <https://www.rfc-editor.org/info/rfc8930>.

[RFC8931] Thubert, P., Ed., "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery", RFC 8931, DOI 10.17487/RFC8931, November 2020, <https://www.rfc-editor.org/info/rfc8931>.

[RIOT] Baccelli, E., Gündoğan, C., Hahm, O., Kietzmann, P., Lenders, MS., Petersen, H., Schleiser, K., Schmidt, TC., and M. Wählisch, "RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT", IEEE Internet of Things Journal Vol. 5, No. 6, p. 4428-4440, December 2018, <https://doi.org/10.1109/JIOT.2018.2815038>.

[SFR-ICNLOWPAN] Lenders, M., Gündoğan, C., Schmidt, TC., and M. Wählisch, "Connecting the Dots: Selective Fragment Recovery in ICNLoWPAN", Proc. of 7th ACM Conf. on Information-Centric Networking (ICN-2020) ACM DL, pp. 70-76, September 2020, <https://doi.org/10.1145/3405656.3418719>.

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

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

Appendix A. Estimated Size Reduction

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

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

A.1. NDN

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

A.1.1. Interest

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

```
          -------------------------------------,
          Interest TLV              = 2         |
            ---------------------,              |
            Name                 |   2 +        |
              NameComponents      = 2n +        |
                                 |   comps_n    |
            ---------------------'               = 21 + 2n + comps_n
            CanBePrefix             = 2         |
            MustBeFresh             = 2         |
            Nonce                   = 6         |
            InterestLifetime        = 4         |
            HopLimit                = 3         |
          -------------------------------------'
```

Figure 34: Estimated Size of an Uncompressed NDN Interest

Figure 35 depicts the size requirements after compression.

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

Figure 35: Estimated Size of a Compressed NDN Interest

The size difference is 11 + 1.5n bytes.

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

A.1.2. Data

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

```
          ------------------------------------,
          Data TLV                 = 2         |
            ---------------------,             |
            Name                 |  2 +        |
              NameComponents      = 2n +       |
                                 |  comps_n    |
            ---------------------'             |
            ---------------------,             |
            MetaInfo             |             |
              FreshnessPeriod     = 6          |
                                 |              = 53 + 2n + comps_n +
            ---------------------'             |  clen + klen
            Content                = 2 + clen  |
            ---------------------,             |
            SignatureInfo        |             |
              SignatureType      |             |
                KeyLocator        = 41 + klen  |
            SignatureValue       |             |
              DigestSha256       |             |
            ---------------------'             |
          ------------------------------------'
```

Figure 36: Estimated Size of an Uncompressed NDN Data

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

```
          ------------------------------------,
          Dispatch Page Switch    = 1         |
          NDN Data Dispatch       = 2         |
          -----------------------,            |
          Name                   |            |
            NameComponents        = n/2 +     |
                                 |  comps_n    = 38 + n/2 + comps_n +
          -----------------------'            |  clen + klen
          Content                 = 1 + clen  |
          KeyLocator              = 1 + klen  |
          DigestSha256            = 32        |
          FreshnessPeriod         = 1         |
          ------------------------------------'
```

Figure 37: Estimated Size of a Compressed NDN Data

The size difference is 15 + 1.5n bytes.

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

A.2. CCNx

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

A.2.1. Interest

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

```
 ------------------------------------,
 Fixed Header              = 8       |
 Message                   = 4       |
   ---------------------,            |
   Name                 |  4 +        = 56 + 4n + comps_n
     NameSegments          = 4n +    |
                        |  comps_n   |
   ---------------------'            |
   KeyIdRestriction        = 40      |
 ------------------------------------'
```

Figure 38: Estimated Size of an Uncompressed CCNx Interest

Figure 39 depicts the size requirements after compression.

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

Figure 39: Estimated Size of a Compressed CCNx Interest

The size difference is 18 + 3.5n bytes.

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

A.2.2. Content Object

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

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

Figure 40: Estimated Size of an Uncompressed CCNx Content Object

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

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

Figure 41: Estimated Size of a Compressed CCNx Data Object

The size difference is 35 + 3.5n bytes.

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

Acknowledgments

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

Authors' Addresses

Cenk Gündoğan
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany
Phone: +4940428758067
Email: cenk.guendogan@haw-hamburg.de
URI: http://inet.haw-hamburg.de/members/cenk-gundogan

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
D-20099 Hamburg
Germany
Email: t.schmidt@haw-hamburg.de
URI: http://inet.haw-hamburg.de/members/schmidt

Matthias Wählisch
link-lab & FU Berlin
Hoenower Str. 35
D-10318 Berlin
Germany
Email: mw@link-lab.net
URI: https://www.mi.fu-berlin.de/en/inf/groups/ilab/members/waehlisch.html

Christopher Scherb
University of Applied Sciences and Arts Northwestern
Switzerland
Peter Merian-Str. 86
CH-4002 Basel
Switzerland
Email: christopher.scherb@fhnw.ch

Claudio Marxer
University of Basel
Spiegelgasse 1
CH-4051 Basel
Switzerland
Email: claudio.marxer@unibas.ch

Christian Tschudin
University of Basel
Spiegelgasse 1
CH-4051 Basel
Switzerland
Email: christian.tschudin@unibas.ch