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

Copyright Notice

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.

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

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.

```
      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 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.

```
             +---+---+---+---+---+---+---+---+
             | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |  Bit field
             +---+---+---+---+---+---+---+---+
               |       |                   |
            ,--'       '-----------,       '- boolean
            |                      |
          +-------+--------------+-------------+
          |  LEN  |     VALUE    |    VALUE    |
          +-------+--------------+-------------+
```

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).

```
    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 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).

```
       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 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.

```
                       0   1   2        ...        7
                     +---+---+-----------------------+
                     | C | M |                       |
                     +---+---+-----------------------+
```

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.

```
+---------------+-----------+--------+---------------+-------+
| IEEE 802.15.4 | Dispatches | Page 2 | NDN Dispatches | Payl. /
+---------------+-----------+--------+---------------+-------+
```

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.

```
+--------------+----------+--------+---------------+------------+
| IEEE 802.15.4 | Frag. 1st | Page 2 |   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 2 |   ICN LoWPAN   | Payload /
+--------------+--------+---------------+---------+
```

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).

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

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.

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

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).

```
       0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-+
   a) |     < 253     |
      +-+-+-+-+-+-+-+-+

       0                   1                   2
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   b) |      253      |      MSB              LSB      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

       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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      254      |      MSB                                      /
   c) +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      LSB      |
      +-+-+-+-+-+-+-+-+

       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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      255      |      MSB                                      /
      +-+-+-+-+-+-+-+-+                                               +
   d) |                                                               /
      +               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      LSB      |
      +-+-+-+-+-+-+-+-+
```

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).

```
       0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-+
   a) |   < 255 (X)   | = X
      +-+-+-+-+-+-+-+-+

       0                   1
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   b) |      255      |   < 255 (X)   | = 255 + X
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

       0
       0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-+-+-+-+-.....-+-+-+-+-+-+-+-+-+-+-+
   c) |      255      |      255      |   < 255 (X)   | = (N * 255) + X
      +-+-+-+-+-+-+-+-+-+-+-+-.....-+-+-+-+-+-+-+-+-+-+-+
                             (N - 1)
```

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

      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 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.

```
  0   1   2         ...         7
+---+---+-----------------------+
| 0 | 0 |         resv          |
+---+---+-----------------------+
```

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.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 |DIG|PFX|FRE|FWD|PRM|CID|
+---+---+---+---+---+---+---+---+
```

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.

```
  0   1   2         ...          7
+---+---+-----------------------+
| 0 | 1 |         resv          |
+---+---+-----------------------+
```

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.

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 1 |DIG|FBI|CON|  SIG  |CID|
+---+---+---+---+---+---+---+---+
```

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.

```
                         0   1   2        ...        7
                       +---+---+----------------------+
                       | 0 | 0 |         resv         |
                       +---+---+----------------------+
```

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.

```
     0                                       1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | 1 | 0 |FLG|HBH|PTY|HPL|FRS|MSG|PAY|VAL|EXT|       RESVD   |CID|
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

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.

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

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

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

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

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

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.

```
  0   1   2         ...         7
+---+---+-----------------------+
| 0 | 1 |         resv          |
+---+---+-----------------------+
```

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.

```
    0                                       1
    0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  | 1 | 1 |FLG|HBH|FRS|MSG|PAY|VAL|EXT|           RESVD       |CID|
  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

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.

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

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

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

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.

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

Figure 28: Estimated size of an uncompressed NDN Interest

Figure 29 depicts the size requirements after compression.

```
        ------------------------------------,
        Dispatch Page Switch    = 1         |
        NDN Interset Dispatch   = 1         |
        Interest TLV            = 1         |
        -----------------------,            |
        Name                   |             = 9 + n/2 + comps_n
          NameComponents        = n/2 +     |
                               |  comps_n   |
        -----------------------'            |
        Nonce                   = 4         |
        InterestLifetime        = 2         |
        ------------------------------------'
```

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.

```
        ------------------------------------,
        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 30: Estimated size of an uncompressed NDN Data

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

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

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.

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

Figure 32: Estimated size of an uncompressed CCNx Interest

Figure 33 depicts the size requirements after compression.

```
------------------------------------,
Dispatch Page Switch    = 1         |
CCNx Interest Dispatch  = 3         |
Fixed Header            = 3         |
-----------------------,            |
Name                   |             = 39 + n/2 + comps_n
  NameSegments          = n/2 +     |
                       |  comps_n   |
-----------------------'            |
T_SHA-256               = 32        |
------------------------------------'
```

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.

```
 ------------------------------------,
 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 34: Estimated size of an uncompressed CCNx Data Object

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

```
 ------------------------------------,
 Dispatch Page Switch    = 1         |
 CCNx Content Dispatch   = 4         |
 Fixed Header            = 2         |
 -----------------------,            |
 Name                   |            |
   NameSegments          = n/2 +     |
                        |  comps_n    = 91 + n/2 + comps_n + clen
 -----------------------'            |
 ExpiryTime              = 8         |
 Payload                 = 1 + clen  |
 T_HMAC-SHA256           = 32        |
 SignatureTime           = 8         |
 ValidationPayload       = 34        |
 ------------------------------------'
```

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