Compression Format for IPv6 Datagrams
over IEEE 802.15.4-Based Networks

Abstract

   This document updates RFC 4944, "Transmission of IPv6 Packets over
   IEEE 802.15.4 Networks". This document specifies an IPv6 header
   compression format for IPv6 packet delivery in Low Power Wireless
   Personal Area Networks (6LoWPANs). The compression format relies on
   shared context to allow compression of arbitrary prefixes. How the
   information is maintained in that shared context is out of scope.
   This document specifies compression of multicast addresses and a
   framework for compressing next headers. UDP header compression is
   specified within this framework.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6282.

Copyright Notice

   Copyright (c) 2011 IETF Trust and the persons identified as the
   document authors. All rights reserved.

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

Table of Contents

1. Introduction

   The [IEEE802.15.4] standard specifies an MTU of 127 bytes, yielding
   about 80 octets of actual Media Access Control (MAC) payload with
   security enabled, on a wireless link with a link throughput of 250
   kbps or less. The 6LoWPAN adaptation format [RFC4944] was specified
   to carry IPv6 datagrams over such constrained links, taking into
   account limited bandwidth, memory, or energy resources that are
   expected in applications such as wireless sensor networks. [RFC4944]
   defines a Mesh Addressing header to support sub-IP forwarding, a
   Fragmentation header to support the IPv6 minimum MTU requirement
   [RFC2460], and stateless header compression for IPv6 datagrams
   (LOWPAN_HC1 and LOWPAN_HC2) to reduce the relatively large IPv6 and
   UDP headers down to (in the best case) several bytes.

   LOWPAN_HC1 and LOWPAN_HC2 are insufficient for most practical uses of
   IPv6 in 6LoWPANs. LOWPAN_HC1 is most effective for link-local
   unicast communication, where IPv6 addresses carry the link-local
   prefix and an Interface Identifier (IID) directly derived from IEEE
   802.15.4 addresses. In this case, both addresses may be completely
   elided. However, though link-local addresses are commonly used for
   local protocol interactions such as IPv6 Neighbor Discovery
   [RFC4861], DHCPv6 [RFC3315], or routing protocols, they are usually
   not used for application-layer data traffic, so the actual value of
   this compression mechanism is limited.

   Routable addresses must be used when communicating with devices
   external to the 6LoWPAN or in a route-over configuration where IP
   forwarding occurs within the 6LoWPAN. For routable addresses,
   LOWPAN_HC1 requires both IPv6 source and destination addresses to
   carry the prefix in-line. In cases where the Mesh Addressing header
   is not used, the IID of a routable address must be carried in-line.
   However, LOWPAN_HC1 requires 64 bits for the IID when carried in-line
   and cannot be shortened even when it is derived from the IEEE
   802.15.4 16-bit short address. When the destination is an IPv6
   multicast address, LOWPAN_HC1 requires the full 128-bit address to be
   carried in-line.

   As a result, this document defines an encoding format, LOWPAN_IPHC,
   for effective compression of Unique Local, Global, and multicast IPv6
   Addresses based on shared state within contexts. In addition, this
   document also introduces a number of additional improvements over the
   header compression format defined in [RFC4944].

   LOWPAN_IPHC allows for compression of some commonly used IPv6 Hop
   Limit values. If the 6LoWPAN is a mesh-under stub, a Hop Limit of 1
   for inbound and a default value such as 64 for outbound are usually
   enough for application-layer data traffic. Additionally, a Hop Limit

value of 255 is often used to verify that a communication occurs over
a single-hop. This specification enables compression of the IPv6 Hop
Limit field in those common cases, whereas LOWPAN_HC1 does not.

This document also defines LOWPAN_NHC, an encoding format for
arbitrary next headers. LOWPAN_IPHC indicates whether the following
header is encoded using LOWPAN_NHC. If so, the bits immediately
following the compressed IPv6 header start the LOWPAN_NHC encoding.
In contrast, LOWPAN_HC1 could be extended to support compression of
next headers using LOWPAN_HC2, but only for UDP, TCP, and ICMPv6.
Furthermore, the LOWPAN_HC2 octet sits between the LOWPAN_HC1 octet
and uncompressed IPv6 header fields. This specification moves the
next header encoding bits to follow all IPv6-related bits, allowing
for a properly layered structure and direct support for IPv6
extension headers.

Using LOWPAN_NHC, this document defines a compression mechanism for
UDP. While [RFC4944] defines a compression mechanism for UDP, that
mechanism does not enable checksum compression when rendered possible
by additional upper-layer mechanisms such as upper-layer Message
Integrity Check (MIC). This specification adds the capability to
elide the UDP checksum over the 6LoWPAN, which enables saving of a
further two octets.

Also, using LOWPAN_NHC, this document defines encoding formats for
IPv6-in-IPv6 encapsulation as well as IPv6 Extension Headers. With
LOWPAN_HC1 and LOWPAN_HC2, chains of next headers cannot be encoded
efficiently.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119].

2. Specific Updates to RFC 4944

This document specifies a header compression format that is intended
to replace that defined in Section 10 of [RFC4944]. Implementation
of Section 10 of [RFC4944] is now NOT RECOMMENDED. New
implementations MAY implement decompression according to Section 10
of [RFC4944] but SHOULD NOT send packets compressed according to
Section 10 of [RFC4944].

A compliant implementation of [RFC4944] as updated by this document
MUST be able to properly process a packet received that makes use of
the provisions of this document. A compliant implementation MAY
implement additional LOWPAN_NHC types (Section 4) that may be

registered (Section 5) in the future. It is out of scope of this
document how a compressor learns that a decompressor has additional
capabilities.

Section 5.3 of [RFC4944] also defines how to fragment compressed IPv6
datagrams that do not fit within a single link frame. Section 5.3 of
[RFC4944] defines the fragment header's datagram_size and
datagram_offset values as the size and offset of the IPv6 datagram
before compression. As a result, all fragment payload outside the
first fragment must carry their respective portions of the IPv6
datagram before compression. This document does not change that
requirement. When using the fragmentation mechanism described in
Section 5.3 of [RFC4944], any header that cannot fit within the first
fragment MUST NOT be compressed.

The header compression format defined in this document preempts the
ESC dispatch value defined in Section 5.1 of [RFC4944]. Instead, the
value of 01 000000 is reserved as a replacement value for ESC, to be
finally assigned with the first assignment of extension bytes.

3. IPv6 Header Compression

In this section, we define the LOWPAN_IPHC encoding format for
compressing the IPv6 header. To enable effective compression,
LOWPAN_IPHC relies on information pertaining to the entire 6LoWPAN.
LOWPAN_IPHC assumes the following will be the common case for 6LoWPAN
communication: Version is 6; Traffic Class and Flow Label are both
zero; Payload Length can be inferred from lower layers from either
the 6LoWPAN Fragmentation header or the IEEE 802.15.4 header; Hop
Limit will be set to a well-known value by the source; addresses
assigned to 6LoWPAN interfaces will be formed using the link-local
prefix or a small set of routable prefixes assigned to the entire
6LoWPAN; addresses assigned to 6LoWPAN interfaces are formed with an
IID derived directly from either the 64-bit extended or the 16-bit
short IEEE 802.15.4 addresses.

```
+-------------------------------------+----------------------------
| Dispatch + LOWPAN_IPHC (2-3 octets) | In-line IPv6 Header Fields
+-------------------------------------+----------------------------
```
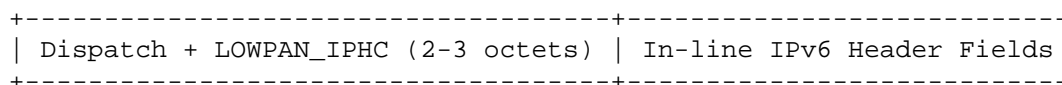
Figure 1: LOWPAN_IPHC Header

The LOWPAN_IPHC encoding utilizes 13 bits, 5 of which are taken from
the rightmost bits of the dispatch type. The encoding may be
extended by another octet to support additional contexts. Any
information from the uncompressed IPv6 header fields carried in-line

follow the LOWPAN_IPHC encoding, as shown in Figure 1. In the best
case, the LOWPAN_IPHC can compress the IPv6 header down to two octets
(the dispatch octet and the LOWPAN_IPHC encoding) with link-local
communication.

When routing over multiple IP hops, LOWPAN_IPHC can compress the IPv6
header down to 7 octets (1-octet dispatch, 1-octet LOWPAN_IPHC,
1-octet Hop Limit, 2-octet Source Address, and 2-octet Destination
Address). The Hop Limit may not be compressed because it needs to
decremented at each hop and may take any value. Stateful address
compression must be applied to the source and destination IPv6
addresses because they do not statelessly match the source and
destination link-layer addresses on intermediate hops.

3.1. LOWPAN_IPHC Encoding Format

   This section specifies the format of the LOWPAN_IPHC encoding that
   describes how an IPv6 header is compressed. The encoding can be 2
   octets long for the base encoding or 3 octets long when an additional
   context encoding is present. The IPv6 header fields that are not
   fully elided are placed immediately after the LOWPAN_IPHC, either in
   a compressed form if the field is partially elided or literally.

3.1.1. Base Format

```
     0                                       1
     0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   | 0 | 1 | 1 |   TF  |NH | HLIM  |CID|SAC|   SAM | M |DAC|  DAM  |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```
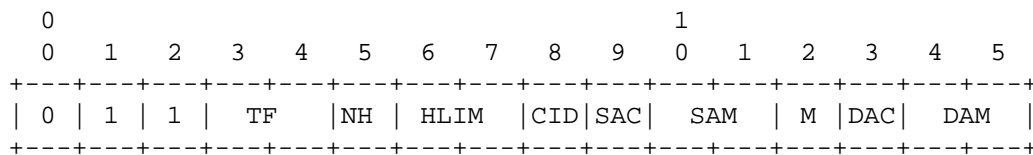
                  Figure 2: LOWPAN_IPHC base Encoding

   TF: Traffic Class, Flow Label: As specified in [RFC3168], the 8-bit
      IPv6 Traffic Class field is split into two fields: 2-bit Explicit
      Congestion Notification (ECN) and 6-bit Differentiated Services
      Code Point (DSCP).

      00: ECN + DSCP + 4-bit Pad + Flow Label (4 bytes)

      01: ECN + 2-bit Pad + Flow Label (3 bytes), DSCP is elided.

      10: ECN + DSCP (1 byte), Flow Label is elided.

      11: Traffic Class and Flow Label are elided.

NH: Next Header:

   0: Full 8 bits for Next Header are carried in-line.

   1: The Next Header field is compressed and the next header is
      encoded using LOWPAN_NHC, which is discussed in Section 4.1.

HLIM: Hop Limit:

   00: The Hop Limit field is carried in-line.

   01: The Hop Limit field is compressed and the hop limit is 1.

   10: The Hop Limit field is compressed and the hop limit is 64.

   11: The Hop Limit field is compressed and the hop limit is 255.

CID: Context Identifier Extension:

   0: No additional 8-bit Context Identifier Extension is used. If
      context-based compression is specified in either Source Address
      Compression (SAC) or Destination Address Compression (DAC),
      context 0 is used.

   1: An additional 8-bit Context Identifier Extension field
      immediately follows the Destination Address Mode (DAM) field.

SAC: Source Address Compression

   0: Source address compression uses stateless compression.

   1: Source address compression uses stateful, context-based
      compression.

SAM: Source Address Mode:

   If SAC=0:

      00: 128 bits. The full address is carried in-line.

      01: 64 bits. The first 64-bits of the address are elided.
         The value of those bits is the link-local prefix padded with
         zeros. The remaining 64 bits are carried in-line.

10: 16 bits. The first 112 bits of the address are elided.
     The value of the first 64 bits is the link-local prefix
     padded with zeros. The following 64 bits are 0000:00ff:
     fe00:XXXX, where XXXX are the 16 bits carried in-line.

11: 0 bits. The address is fully elided. The first 64 bits
     of the address are the link-local prefix padded with zeros.
     The remaining 64 bits are computed from the encapsulating
     header (e.g., 802.15.4 or IPv6 source address) as specified
     in Section 3.2.2.

If SAC=1:

00: The UNSPECIFIED address, ::

01: 64 bits. The address is derived using context information
     and the 64 bits carried in-line. Bits covered by context
     information are always used. Any IID bits not covered by
     context information are taken directly from the
     corresponding bits carried in-line. Any remaining bits are
     zero.

10: 16 bits. The address is derived using context information
     and the 16 bits carried in-line. Bits covered by context
     information are always used. Any IID bits not covered by
     context information are taken directly from their
     corresponding bits in the 16-bit to IID mapping given by
     0000:00ff:fe00:XXXX, where XXXX are the 16 bits carried in-
     line. Any remaining bits are zero.

11: 0 bits. The address is fully elided and is derived using
     context information and the encapsulating header (e.g.,
     802.15.4 or IPv6 source address). Bits covered by context
     information are always used. Any IID bits not covered by
     context information are computed from the encapsulating
     header as specified in Section 3.2.2. Any remaining bits
     are zero.

M: Multicast Compression

   0: Destination address is not a multicast address.

   1: Destination address is a multicast address.

   DAC: Destination Address Compression

      0: Destination address compression uses stateless compression.

      1: Destination address compression uses stateful, context-based
         compression.

   DAM: Destination Address Mode:

      If M=0 and DAC=0 This case matches SAC=0 but for the destination
         address:

         00: 128 bits. The full address is carried in-line.

         01: 64 bits. The first 64-bits of the address are elided.
            The value of those bits is the link-local prefix padded with
            zeros. The remaining 64 bits are carried in-line.

         10: 16 bits. The first 112 bits of the address are elided.
            The value of the first 64 bits is the link-local prefix
            padded with zeros. The following 64 bits are 0000:00ff:
            fe00:XXXX, where XXXX are the 16 bits carried in-line.

         11: 0 bits. The address is fully elided. The first 64 bits
            of the address are the link-local prefix padded with zeros.
            The remaining 64 bits are computed from the encapsulating
            header (e.g., 802.15.4 or IPv6 destination address) as
            specified in Section 3.2.2.

      If M=0 and DAC=1:

         00: Reserved.

         01: 64 bits. The address is derived using context information
            and the 64 bits carried in-line. Bits covered by context
            information are always used. Any IID bits not covered by
            context information are taken directly from the
            corresponding bits carried in-line. Any remaining bits are
            zero.

         10: 16 bits. The address is derived using context information
            and the 16 bits carried in-line. Bits covered by context
            information are always used. Any IID bits not covered by
            context information are taken directly from their
            corresponding bits in the 16-bit to IID mapping given by
            0000:00ff:fe00:XXXX, where XXXX are the 16 bits carried in-
            line. Any remaining bits are zero.

11: 0 bits. The address is fully elided and is derived using
     context information and the encapsulating header (e.g.
     802.15.4 or IPv6 destination address). Bits covered by
     context information are always used. Any IID bits not
     covered by context information are computed from the
     encapsulating header as specified in Section 3.2.2. Any
     remaining bits are zero.

If M=1 and DAC=0:

   00: 128 bits. The full address is carried in-line.

   01: 48 bits. The address takes the form ffXX::00XX:XXXX:XXXX.

   10: 32 bits. The address takes the form ffXX::00XX:XXXX.

   11: 8 bits. The address takes the form ff02::00XX.

If M=1 and DAC=1:

   00: 48 bits. This format is designed to match Unicast-Prefix-
        based IPv6 Multicast Addresses as defined in [RFC3306] and
        [RFC3956]. The multicast address takes the form ffXX:XXLL:
        PPPP:PPPP:PPPP:PPPP:XXXX:XXXX. where the X are the nibbles
        that are carried in-line, in the order in which they appear
        in this format. P denotes nibbles used to encode the prefix
        itself. L denotes nibbles used to encode the prefix length.
        The prefix information P and L is taken from the specified
        context.

   01: reserved

   10: reserved

   11: reserved

3.1.2. Context Identifier Extension

   This specification expects that a conceptual context is shared
   between the node that compresses a packet and the node(s) that needs
   to expand it. How the contexts are shared and maintained is out of
   scope. What information is contained within a context information is
   out of scope. Actions in response to unknown and/or invalid contexts
   are out of scope. The specification enables a node to use up to 16
   contexts. The context used to encode the source address does not
   have to be the same as the context used to encode the destination
   address.

If the CID field is set to '1' in the LOWPAN_IPHC encoding, then an
additional octet extends the LOWPAN_IPHC encoding following the DAM
bits but before the IPv6 header fields that are carried in-line. The
additional octet identifies the pair of contexts to be used when the
IPv6 source and/or destination address is compressed. The context
identifier is 4 bits for each address, supporting up to 16 contexts.
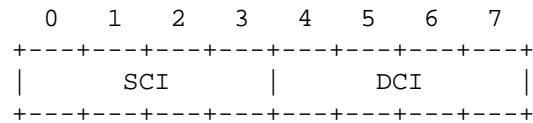Context 0 is the default context. The encoding is shown in Figure 3.

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

Figure 3: LOWPAN_IPHC Encoding

SCI: Source Context Identifier. Identifies the prefix that is used
   when the IPv6 source address is statefully compressed.

DCI: Destination Context Identifier. Identifies the prefix that is
   used when the IPv6 destination address is statefully compressed.

3.2. IPv6 Header Encoding

Fields carried in-line (in part or in whole) appear in the same order
as they do in the IPv6 header format [RFC2460]. The Version field is
always elided. Unicast IPv6 addresses may be compressed to 64 or 16
bits or completely elided. Multicast IPv6 addresses may be
compressed to 8, 32, or 48 bits. The IPv6 Payload Length field MUST
always be elided and inferred from lower layers using the 6LoWPAN
Fragmentation header or the IEEE 802.15.4 header.

3.2.1. Traffic Class and Flow Label Compression

The Traffic Class field in the IPv6 header comprises 6 bits of
Diffserv extension [RFC2474] and 2 bits of Explicit Congestion
Notification (ECN) [RFC3168]. The TF field in the LOWPAN_IPHC
encoding indicates whether the Traffic Class and Flow Label are
carried in-line in the compressed IPv6 header. When Flow Label
included while the Traffic Class is compressed, an additional 4 bits
are included to maintain byte alignment. Two of the 4 bits contain
the ECN bits from the Traffic Class field.

To ensure that the ECN bits appear in the same location for all
encodings that include them, the Traffic Class field is rotated right
by 2 bits in the compressed IPv6 header. The encodings are shown
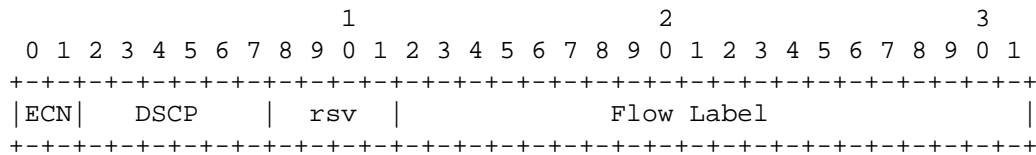below:

```
                          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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |ECN|   DSCP    | rsv |                 Flow Label                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

     Figure 4: TF = 00: Traffic Class and Flow Label carried in-line

```
                          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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |ECN|rsv|              Flow Label                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
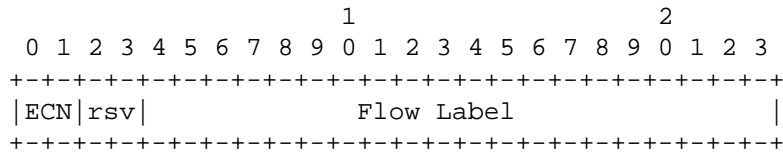
             Figure 5: TF = 01: Flow Label carried in-line

```
    0 1 2 3 4 5 6 7
   +-+-+-+-+-+-+-+-+
   |ECN|   DSCP    |
   +-+-+-+-+-+-+-+-+
```
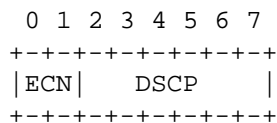
           Figure 6: TF = 10: Traffic Class carried in-line

3.2.2. Deriving IIDs from the Encapsulating Header

   LOWPAN_IPHC elides the IIDs of source or destination addresses when
   SAM = 3 or DAM = 3, respectively. In this mode, the IID is derived
   from the encapsulating header. When the encapsulating header carries
   IPv6 addresses, bits for the source and destination addresses are
   copied from the source and destination addresses of the encapsulating
   IPv6 header.

   The remainder of this section defines the mapping from IEEE 802.15.4
   [IEEE802.15.4] link-layer addresses to IIDs for both short and
   extended IEEE 802.15.4 addresses. IID bits not covered by the
   context information MAY be elided if they match the link-layer
   address mapping and MUST NOT be elided if they do not.

   An extended IEEE 802.15.4 address takes the form of an IEEE EUI-64
   address. Generating an IID from an extended address is identical to
   that defined in Appendix A of [RFC4291]. The only change needed to
   transform an IEEE EUI-64 identifier to an interface identifier is to
   invert the universal/local bit.

A short IEEE 802.15.4 address is 16 bits in length. Short addresses
are mapped into the restricted space of IEEE EUI-64 addresses by
setting the middle 16 bits to 0xfffe, the bottom 16 bits to the short
address, and all other bits to zero. As a result, an IID generated
from a short address has the form:

      0000:00ff:fe00:XXXX

where XXXX carries the short address. The universal/local bit is
zero to indicate local scope.

This mapping for non-EUI-64 identifiers differs from that presented
in Appendix A of [RFC4291]. Using the restricted space ensures no
overlap with IIDs generated from unrestricted IEEE EUI-64 addresses.
Also, including 0xfffe in the middle of the IID helps avoid overlap
with other locally managed IIDs.

This mapping from a short IEEE 802.15.4 address to 64-bit IIDs is
also used to reconstruct any part of an IID not covered by context
information.

3.2.3. Stateless Multicast Address Compression

   LOWPAN_IPHC supports stateless compression of multicast addresses
   when M = 1 and DAC = 0. An IPv6 multicast address may be compressed
   down to 48, 32, or 8 bits using stateless compression. The format
   supports compression of the Solicited-Node Multicast Address (ff02::
   1:ffXX:XXXX) as well as any IPv6 multicast address where the upper
   bits of the multicast group identifier are zero. The 8-bit
   compressed form only carries the least-significant bits of the
   multicast group identifier. The 48- and 32-bit compressed forms
   carry the multicast scope and flags in-line, in addition to the
   least-significant bits of the multicast group identifier.

```
                       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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  | Flags | Scope |                 Group Identifier              |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Group Identifier        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
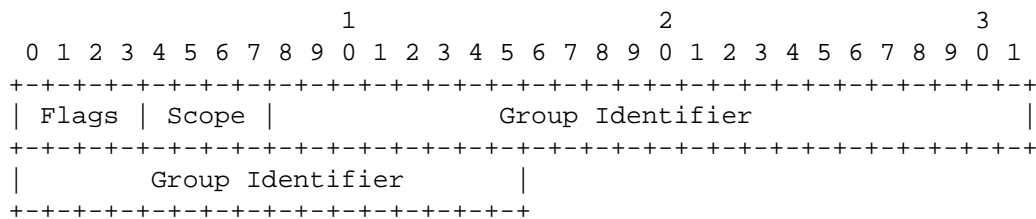
           Figure 7: DAM = 01. 48-bit Compressed Multicast Address
                           (ffFS::00GG:GGGG:GGGG)

```
                             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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Flags | Scope |                Group Identifier               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
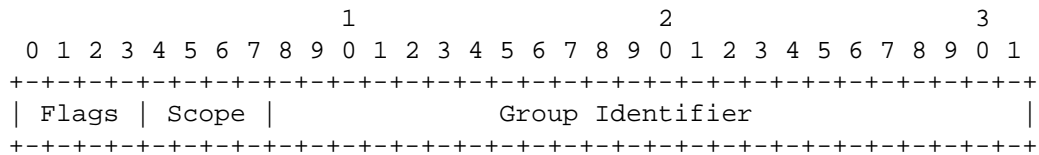
        Figure 8: DAM = 10. 32-bit Compressed Multicast Address
                           (ffFS::00GG:GGGG)

```
     0 1 2 3 4 5 6 7
    +-+-+-+-+-+-+-+-+
    |   Group ID    |
    +-+-+-+-+-+-+-+-+
```
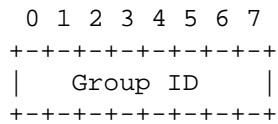
     Figure 9: DAM = 11. 8-bit Compressed Multicast Address (ff02::GG)

3.2.4. Stateful Multicast Address Compression

   LOWPAN_IPHC supports stateful compression of multicast addresses when
   M = 1 and DAC = 1. This document currently defines DAM = 00:
   context-based compression of Unicast-Prefix-based IPv6 Multicast
   Addresses [RFC3306][RFC3956]. In particular, the Prefix Length and
   Network Prefix can be taken from a context. As a result, LOWPAN_IPHC
   can compress a Unicast-Prefix-based IPv6 Multicast Address down to 6
   octets by only carrying the 4-bit Flags, 4-bit Scope, 8-bit
   Rendezvous Point Interface ID (RIID), and 32-bit Group Identifier in-
   line.

```
                             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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Flags | Scope | Rsvd / RIID   |        Group Identifier       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Group Identifier     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
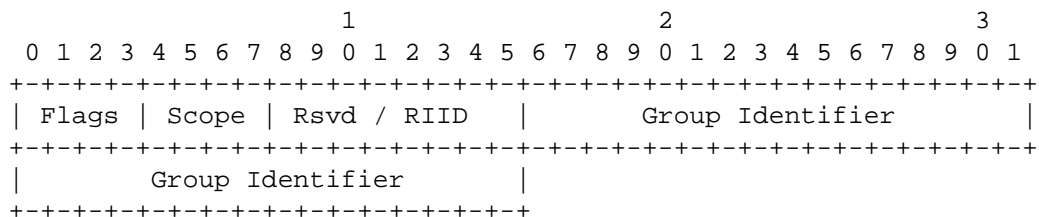
        Figure 10: DAM = 00. Unicast-Prefix-based IPv6 Multicast
                          Address Compression

   Note that the Reserved field MUST carry the reserved bits from the
   multicast address format as described in [RFC3306]. When a
   Rendezvous Point is encoded in the multicast address as described in
   [RFC3956], the Reserved field carries the RIID bits in-line.

4. IPv6 Next Header Compression

   LOWPAN_IPHC elides the IPv6 Next Header field when the NH bit is set
   to 1. This also indicates the use of 6LoWPAN next header
   compression, LOWPAN_NHC. The value of IPv6 Next Header is recovered
   from the first bits in the LOWPAN_NHC encoding. The following bits
   are specific to the IPv6 Next Header value. Figure 11 shows the
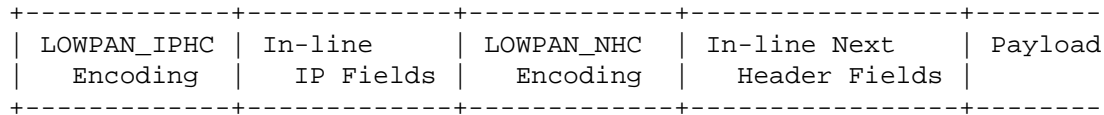   structure of an IPv6 datagram compressed using LOWPAN_IPHC and
   LOWPAN_NHC.

```
   +-------------+-------------+-------------+-----------------+--------
   | LOWPAN_IPHC | In-line     | LOWPAN_NHC  | In-line Next    | Payload
   |   Encoding  |    IP Fields|   Encoding  |   Header Fields |
   +-------------+-------------+-------------+-----------------+--------
```

        Figure 11: Typical LOWPAN_IPHC/LOWPAN_NHC Header Configuration

4.1. LOWPAN_NHC Format

   Compression formats for different next headers are identified by a
   variable-length bit-pattern immediately following the LOWPAN_IPHC
   compressed header. When defining a next header compression format,
   the number of bits used SHOULD be determined by the perceived
   frequency of using that format. However, the number of bits and any
   remaining encoding bits SHOULD respect octet alignment. The
   following bits are specific to the next header compression format.
   This document defines a compression format for IPv6 Extension and UDP
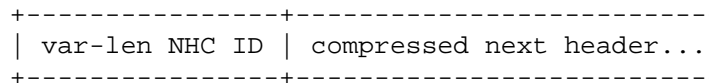   headers.

```
              +----------------+--------------------------
              | var-len NHC ID | compressed next header...
              +----------------+--------------------------
```

                      Figure 12: LOWPAN_NHC Encoding

4.2. IPv6 Extension Header Compression

   A necessary property of encoding headers using LOWPAN_NHC is that the
   immediately preceding header must be encoded using either LOWPAN_IPHC
   or LOWPAN_NHC. In other words, all headers encoded using the 6LoWPAN
   encoding format defined in this document must be contiguous. As a
   result, this document defines a set of LOWPAN_NHC encodings for
   selected IPv6 Extension Headers such that the UDP Header Compression
   defined in Section 4.3 may be used in the presence of those extension
   headers.

The LOWPAN_NHC encodings for IPv6 Extension Headers are composed of a
single LOWPAN_NHC octet followed by the IPv6 Extension Header. The
format of the LOWPAN_NHC octet is shown in Figure 13. The first 7
bits serve as an identifier for the IPv6 Extension Header immediately
following the LOWPAN_NHC octet. The remaining bit indicates whether
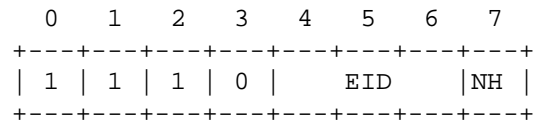or not the following header utilizes LOWPAN_NHC encoding.

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

                Figure 13: IPv6 Extension Header Encoding

EID: IPv6 Extension Header ID:

   0: IPv6 Hop-by-Hop Options Header [RFC2460]

   1: IPv6 Routing Header [RFC2460]

   2: IPv6 Fragment Header [RFC2460]

   3: IPv6 Destination Options Header [RFC2460]

   4: IPv6 Mobility Header [RFC6275]

   5: Reserved

   6: Reserved

   7: IPv6 Header

NH: Next Header:

   0: Full 8 bits for Next Header are carried in-line.

   1: The Next Header field is elided and the next header is encoded
      using LOWPAN_NHC, which is discussed in Section 4.1.

For the most part, the IPv6 Extension Header is carried unmodified in
the bytes immediately following the LOWPAN_NHC octet, with two
important exceptions: Length field and Next Header field.

The Next Header field contained in IPv6 Extension Headers is elided
when the NH bit is set in the LOWPAN_NHC encoding octet. Note that
doing so allows LOWPAN_NHC to utilize no more overhead than the non-
encoded IPv6 Extension Header.

The Length field contained in a compressed IPv6 Extension Header
indicates the number of octets that pertain to the (compressed)
extension header following the Length field. Note that this changes
the Length field definition in [RFC2460] from indicating the header
size in 8-octet units, not including the first 8 octets. Changing
the Length field to be in units of octets removes wasteful internal
fragmentation.

IPv6 Hop-by-Hop and Destination Options Headers may use a trailing
Pad1 or PadN to achieve 8-octet alignment. When there is a single
trailing Pad1 or PadN option of 7 octets or less and the containing
header is a multiple of 8 octets, the trailing Pad1 or PadN option
MAY be elided by the compressor. A decompressor MUST ensure that the
containing header is padded out to a multiple of 8 octets in length,
using a Pad1 or PadN option if necessary. Note that Pad1 and PadN
options that appear in locations other than the end MUST be carried
in-line as they are used to align subsequent options.

Note that specifying units in octets means that LOWPAN_NHC MUST NOT
be used to encode IPv6 Extension Headers that have more than 255
octets following the Length field after compression.

When the identified next header is an IPv6 Header (EID=7), the NH bit
of the LOWPAN_NHC encoding is unused and MUST be set to zero. The
following bytes MUST be encoded using LOWPAN_IPHC as defined in
Section 3.

4.3. UDP Header Compression

This document defines a compression format for UDP headers using
LOWPAN_NHC. The UDP compression format is shown in Figure 14. Bits
0 through 4 represent the NHC ID and '11110' indicates the specific
UDP header compression encoding defined in this section.

4.3.1. Compressing UDP Ports

This specification allows a particular range of ports number (0xf0b0
to 0xf0bf) to be compressed down to 4 bits. This is a stateless
compression that is inherited from [RFC4944], as opposed to a new
stateful compression.

The range of ports compressible down to 4 bits is not in a reserved
range. A network stack implementation that is designed to
communicate over a 6LoWPAN should avoid using those ports as dynamic
ports whenever possible.

Considering that this represents only 16 contiguous ports, it can be
expected that many incompatible applications will use the same value
of port numbers for their own end-to-end needs. Thus, a port number
in the (0xf0b0 to 0xf0bf) range provides very little information
about the application at the remote end.

The overloading of the 0xf0bX ports increases the risk of getting the
wrong type of payload and misinterpreting the content compared to
ports that are reserved at IANA. As a result, it is recommended that
the use of those ports be associated with a mechanism such as a
Transport Layer Security (TLS) [RFC5246] Message Integrity Check
(MIC) that makes sure that the content is what is expected and is
checked.

4.3.2. Compressing UDP Checksum

The UDP checksum operation is mandatory with IPv6 [RFC2460] for all
packets. For that reason, [RFC4944] disallows the compression of the
UDP checksum.

With this specification, a compressor in the source transport
endpoint MAY elide the UDP Checksum if it is authorized by the upper
layer. The compressor MUST NOT set the C bit unless it has received
such authorization. Requiring upper-layer authorization ensures that
the intended transport peer will have sufficient means to deal with
any data corruption that occurs before reaching the destination. The
upper layer MUST NOT provide the authorization unless one of the
following cases is satisfied:

Tunneling: In this case, 6LoWPAN is deployed as a wireless pseudo-
   fieldbus by tunneling existing field protocols over UDP. If the
   tunneled Protocol Data Unit (PDU) possesses its own addressing,
   security and integrity check (e.g., IPsec Encapsulating Security
   Payload tunnel mode [RFC4303] or IP over UDP encapsulation), the
   tunneling mechanism MAY authorize eliding the UDP checksum in
   order to save on the encapsulation overhead.

Message Integrity Check: In this case, either IPsec Authentication
   Header [RFC4302] or some other form of integrity check in the UDP
   payload that covers at least the same information as the UDP
   checksum (pseudo-header, data) and has at least the same strength.

To help ensure that the UDP Checksum will be properly restored when
expanding a 6LoWPAN packet, an additional integrity check (e.g., a
Layer 2 (L2) Message Integrity Check) MUST be used whenever
transmitting link frames that carry a compressed UDP datagram that

elides the checksum. Without this additional integrity check, a UDP
packet may be delivered to an unintended destination since corruption
in data covered by the pseudo-header can go undetected.

A compressor MUST verify the UDP Checksum before it is elided and
MUST ensure that the additional integrity check is in place before
verifying and eliding the checksum. If verification of the UDP
Checksum fails, the compressor MUST drop the packet.

A decompressor that expands a 6LoWPAN packet with the C bit set MUST
compute the UDP Checksum on behalf of the source node and place that
value in the restored UDP header as specified in the incumbent
standards [RFC0768], [RFC2460]. The decompressor MUST unambiguously
determine that an additional integrity check was put in place by the
compressor and verify the integrity check and SHOULD do so after
restoring the UDP Checksum. If the decompressor cannot unambiguously
determine the presence of an integrity check or verification fails,
the decompressor MUST drop the packet.

The recommended ordering of computing and verifying the UDP Checksum
and additional integrity check ensures that data is never stored
unprotected in memory. In practice, functionality separation between
layers may preclude the recommended ordering. However, implementors
should take special note and understand the risks when dealing with
unprotected data covered by the pseudo-header.

To allow intermediate nodes to compress the UDP Checksum, a
forwarding node MAY infer upper-layer authorization for an incoming
packet if it has the C bit set and it can unambiguously determine
that an integrity check covering the same data as the UDP Checksum
was in place while the UDP Checksum was elided. A forwarding node
MUST NOT infer authorization if it cannot unambiguously determine the
presence of and verify an integrity check while the UDP Checksum was
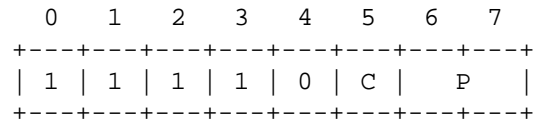elided.

4.3.3. UDP LOWPAN_NHC Format

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

                   Figure 14: UDP Header Encoding

   C: Checksum:

      0: All 16 bits of Checksum are carried in-line.

      1: All 16 bits of Checksum are elided. The Checksum is recovered
         by recomputing it on the 6LoWPAN termination point.

   P: Ports:

      00: All 16 bits for both Source Port and Destination Port are
         carried in-line.

      01: All 16 bits for Source Port are carried in-line. First 8
         bits of Destination Port is 0xf0 and elided. The remaining 8
         bits of Destination Port are carried in-line.

      10: First 8 bits of Source Port are 0xf0 and elided. The
         remaining 8 bits of Source Port are carried in-line. All 16
         bits for Destination Port are carried in-line.

      11: First 12 bits of both Source Port and Destination Port are
         0xf0b and elided. The remaining 4 bits for each are carried
         in-line.

   Fields carried in-line (in part or in whole) appear in the same order
   as they do in the UDP header format [RFC0768]. The UDP Length field
   MUST always be elided and is inferred from lower layers using the
   6LoWPAN Fragmentation header or the IEEE 802.15.4 header.

5. IANA Considerations

   This document defines a new IPv6 header compression format for
   6LoWPAN. The document allocates the following 32 Dispatch type field
   values for LOWPAN_IPHC:

      01 100000
       through
      01 111111

This assignment preempts the assignment of 01 111111 for ESC
[RFC4944]; this preemption is possible because extension bytes that
would enable the use of ESC have not been allocated yet. Instead,
the value:

   01 000000

is reserved as a replacement value for ESC, to be finally assigned
with the first assignment of extension bytes.

This document also creates a new IANA registry for the LOWPAN_NHC
header type, with the following initial content:

```
00000000 to 11011111: (unassigned)
1110000N: IPv6 Hop-by-Hop Options Header       [RFC6282]
1110001N: IPv6 Routing Header                  [RFC6282]
1110010N: IPv6 Fragment Header                 [RFC6282]
1110011N: IPv6 Destination Options Header      [RFC6282]
1110100N: IPv6 Mobility Header                 [RFC6282]
1110111N: IPv6 Header                          [RFC6282]
11110CPP: UDP Header                           [RFC6282]
11111000 to 11111110: (unassigned)
```

Capital letters in bit positions represent class-specific bit
assignments. N indicates whether or not additional LOWPAN_NHC
encodings follow, as defined in Section 4.2. CPP represents
variables specific to UDP header compression defined in Section 4.3.

The policy for this registry [RFC5226] is IETF Review. In this
process, new values SHOULD be assigned in a way that preserves the
NHC ID abstraction of Section 4 (i.e., k one-bits followed by one
zero-bit identify the general class of NHC, followed by class-
specific bit assignments).

6. Security Considerations

The definition of LOWPAN_IPHC permits the compression of header
information on communication that could take place in its absence,
albeit in a less efficient form. It recognizes that a IEEE 802.15.4
PAN may have associated with it a number of prefixes through shared
context. How the shared context is assigned and managed is beyond
the scope of this document.

The overloading of the 0xf0bX ports increases the risk of getting the
wrong type of payload and misinterpreting the content compared to
ports that reserved at IANA. It is thus recommended that the use of

those ports be associated with a mechanism such as a Transport Layer
Security (TLS) [RFC5246] Message Integrity Check (MIC) that validates
that the content is expected and checked for integrity.

7. Acknowledgements

Thanks to Julien Abeille, Robert Assimiti, Dominique Barthel, Carsten
Bormann, Robert Cragie, Stephen Dawson-Haggerty, Mathilde Durvy, Erik
Nordmark, Christos Polyzois, Joseph Reddy, Shoichi Sakane, Zach
Shelby, Dario Tedeschi, Tony Viscardi, and Jay Werb for useful design
consideration and implementation feedback. Special thanks to David
Black, Lars Eggert, and Carsten Bormann for their contribution in
closing the security issues around UDP compression.

8. References

8.1. Normative References

[RFC0768]   Postel, J., "User Datagram Protocol", STD 6, RFC 768,
               August 1980.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460]   Deering, S. and R. Hinden, "Internet Protocol,
               Version 6 (IPv6) Specification", RFC 2460,
               December 1998.

[RFC2474]   Nichols, K., Blake, S., Baker, F., and D. Black,
               "Definition of the Differentiated Services Field (DS
               Field) in the IPv4 and IPv6 Headers", RFC 2474,
               December 1998.

[RFC3168]   Ramakrishnan, K., Floyd, S., and D. Black, "The
               Addition of Explicit Congestion Notification (ECN) to
               IP", RFC 3168, September 2001.

[RFC4291]   Hinden, R. and S. Deering, "IP Version 6 Addressing
               Architecture", RFC 4291, February 2006.

[RFC4944]   Montenegro, G., Kushalnagar, N., Hui, J., and D.
               Culler, "Transmission of IPv6 Packets over IEEE
               802.15.4 Networks", RFC 4944, September 2007.

[RFC5226]   Narten, T. and H. Alvestrand, "Guidelines for Writing
               an IANA Considerations Section in RFCs", BCP 26,
               RFC 5226, May 2008.

   [RFC6275]   Perkins, C., Ed., Johnson, D., and J. Arkko,
                    "Mobility Support in IPv6", RFC 6275, July 2011.

8.2. Informative References

   [IEEE802.15.4] IEEE Computer Society, "IEEE Std. 802.15.4-2006",
                    October 2006.

   [RFC3306]   Haberman, B. and D. Thaler, "Unicast-Prefix-based
                    IPv6 Multicast Addresses", RFC 3306, August 2002.

   [RFC3315]   Droms, R., Bound, J., Volz, B., Lemon, T., Perkins,
                    C., and M. Carney, "Dynamic Host Configuration
                    Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

   [RFC3956]   Savola, P. and B. Haberman, "Embedding the Rendezvous
                    Point (RP) Address in an IPv6 Multicast Address",
                    RFC 3956, November 2004.

   [RFC4302]   Kent, S., "IP Authentication Header", RFC 4302,
                    December 2005.

   [RFC4303]   Kent, S., "IP Encapsulating Security Payload (ESP)",
                    RFC 4303, December 2005.

   [RFC4861]   Narten, T., Nordmark, E., Simpson, W., and H.
                    Soliman, "Neighbor Discovery for IP version 6
                    (IPv6)", RFC 4861, September 2007.

   [RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer
                    Security (TLS) Protocol Version 1.2", RFC 5246,
                    August 2008.

Authors' Addresses

    Jonathan W. Hui (editor)
    Arch Rock Corporation
    501 2nd St. Ste. 410
    San Francisco, California 94107
    USA

    Phone: +415 692 0828
    EMail: jhui@archrock.com

    Pascal Thubert
    Cisco Systems
    Village d'Entreprises Green Side
    400, Avenue de Roumanille
    Batiment T3
    Biot - Sophia Antipolis 06410
    FRANCE

    Phone: +33 4 97 23 26 34
    EMail: pthubert@cisco.com