

6Lo Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 21, 2014

C. Bormann  
Universitaet Bremen TZI  
October 18, 2013

6LoWPAN Generic Compression of Headers and Header-like Payloads  
draft-bormann-6lo-ghc-00

Abstract

This short I-D provides a simple addition to 6LoWPAN Header Compression that enables the compression of generic headers and header-like payloads, without a need to define a new header compression scheme for each new such header or header-like payload.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2014.

Copyright Notice

Copyright (c) 2013 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 . . . . .	2
1.1. The Header Compression Coupling Problem . . . . .	2
1.2. Terminology . . . . .	2
1.3. Notation . . . . .	3
2. 6LoWPAN-GHC . . . . .	4
3. Integrating 6LoWPAN-GHC into 6LoWPAN-HC . . . . .	5
3.1. Compressing payloads (UDP and ICMPv6) . . . . .	5
3.2. Compressing extension headers . . . . .	5
3.3. Indicating GHC capability . . . . .	6
3.4. Using the 6CIO Option . . . . .	7
4. IANA considerations . . . . .	8
5. Security considerations . . . . .	9
6. Acknowledgements . . . . .	9
7. References . . . . .	10
7.1. Normative References . . . . .	10
7.2. Informative References . . . . .	10
Appendix A. Examples . . . . .	11
Author's Address . . . . .	21

## 1. Introduction

## 1.1. The Header Compression Coupling Problem

6LoWPAN-HC [RFC6282] defines a scheme for header compression in 6LoWPAN [RFC4944] packets. As with most header compression schemes, a new specification is needed for every new kind of header that needs to be compressed. In addition, [RFC6282] does not define an extensibility scheme like the ROHC profiles defined in ROHC [RFC3095] [RFC5795]. This leads to the difficult situation that 6LoWPAN-HC tended to be reopened and reexamined each time a new header receives consideration (or an old header is changed and reconsidered) in the 6LoWPAN/roll/CoRE cluster of IETF working groups. While [RFC6282] finally got completed, the underlying problem remains unsolved.

The purpose of the present contribution is to plug into [RFC6282] as is, using its NHC (next header compression) concept. We add a slightly less efficient, but vastly more general form of compression for headers of any kind and even for header-like payloads such as those exhibited by routing protocols, DHCP, etc. The objective is an extremely simple specification that can be defined on a single page and implemented in a small number of lines of code, as opposed to a general data compression scheme such as that defined in [RFC1951].

## 1.2. 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 term "byte" is used in its now customary sense as a synonym for "octet".

### 1.3. Notation

This specification uses a trivial notation for code bytes and the bitfields in them the meaning of which should be mostly obvious. More formally speaking, the meaning of the notation is:

Potential values for the code bytes themselves are expressed by templates that represent 8-bit most-significant-bit-first binary numbers (without any special prefix), where 0 stands for 0, 1 for 1, and variable segments in these code byte templates are indicated by sequences of the same letter such as kkkkkkkk or ssss, the length of which indicates the length of the variable segment in bits.

In the notation of values derived from the code bytes, 0b is used as a prefix for expressing binary numbers in most-significant-bit first notation (akin to the use of 0x for most-significant-digit-first hexadecimal numbers in the C programming language). Where the above-mentioned sequences of letters are then referenced in such a binary number in the text, the intention is that the value from these bitfields in the actual code byte be inserted.

Example: The code byte template

101nssss

stands for a byte that starts (most-significant-bit-first) with the bits 1, 0, and 1, and continues with five variable bits, the first of which is referenced as "n" and the next four are referenced as "ssss". Based on this code byte template, a reference to

0b0sssss000

means a binary number composed from a zero bit, the four bits that are in the "ssss" field (for 101nssss, the four least significant bits) in the actual byte encountered, kept in the same order, and three more zero bits.

## 2. 6LoWPAN-GHC

The format of a GHC-compressed header or payload is a simple bytecode. A compressed header consists of a sequence of pieces, each of which begins with a code byte, which may be followed by zero or more bytes as its argument. Some code bytes cause bytes to be laid out in the destination buffer, some simply modify some decompression variables.

At the start of decompressing a header or payload within a L2 packet (= fragment), variables "sa" and "na" are initialized as zero.

The code bytes are defined as follows (Table 1):

code byte	Action	Argument
0kkkkkkk	Append k = 0b0kkkkkkk bytes of data in the bytecode argument (k < 96)	k bytes of data
1000nnnn	Append 0b0000nnnn+2 bytes of zeroes	
10010000	STOP code (end of compressed data, see Section 3.2)	
101nssss	Set up extended arguments for a backreference: sa += 0b0ssss000, na += 0b0000n000	
11nnnkkk	Backreference: n = na+0b00000nnn+2; s = 0b00000kkk+sa+n; append n bytes from previously output bytes, starting s bytes to the left of the current output pointer; set sa = 0, na = 0	

Table 1: Bytecodes for generic header compression

Note that the following bit combinations are reserved at this time: 011xxxxx, and 1001nnnn (where 0b0000nnnn > 0).

For the purposes of the backreferences, the expansion buffer is initialized with a predefined dictionary, at the end of which the target buffer begins. This dictionary is composed of the pseudo-header for the current packet as defined in [RFC2460], followed by a 16-byte static dictionary (Figure 1). These dictionary bytes are therefore available for backreferencing, but not copied into the final result.

```
16 fe fd 17 fe fd 00 01 00 00 00 00 00 01 00 00
```

Figure 1: The 16 bytes of static dictionary (in hex)

### 3. Integrating 6LoWPAN-GHC into 6LoWPAN-HC

6LoWPAN-GHC plugs in as an NHC format for 6LoWPAN-HC [RFC6282].

#### 3.1. Compressing payloads (UDP and ICMPv6)

GHC is by definition generic and can be applied to different kinds of packets. Many of the examples given in Appendix A are for ICMPv6 packets; a single NHC value suffices to define an NHC format for ICMPv6 based on GHC (see below).

In addition it is useful to include an NHC format for UDP, as many headerlike payloads (e.g., DHCPv6, DTLS) are carried in UDP. [RFC6282] already defines an NHC format for UDP (11110CPP). GHC uses an analogous NHC byte formatted as shown in Figure 2. The difference to the existing UDP NHC specification is that for 0b11010cpp NHC bytes, the UDP payload is not supplied literally but compressed by 6LoWPAN-GHC.

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

Figure 2: NHC byte for UDP GHC (to be allocated by IANA)

To stay in the same general numbering space, we use 0b11011111 as the NHC byte for ICMPv6 GHC (Figure 3).

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

Figure 3: NHC byte for ICMPv6 GHC (to be allocated by IANA)

#### 3.2. Compressing extension headers

Compression of specific extension headers is added in a similar way (Figure 4) (however, probably only EID 0 to 3 need to be assigned). As there is no easy way to extract the length field from the GHC-encoded header before decoding, this would make detecting the end of the extension header somewhat complex. The easiest (and most efficient) approach is to completely elide the length field (in the

same way NHC already elides the next header field in certain cases) and reconstruct it only on decompression. To serve as a terminator for the extension header, the reserved bytecode 0b10010000 has been assigned as a stop marker. Note that the stop marker is only needed for extension headers, not for the final payloads discussed in the previous subsection, the decompression of which is automatically stopped by the end of the packet.

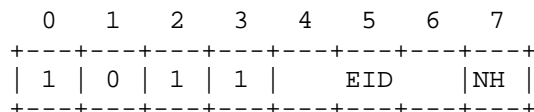


Figure 4: NHC byte for extension header GHC

### 3.3. Indicating GHC capability

The 6LoWPAN baseline includes just [RFC4944], [RFC6282], [RFC6775] (see [I-D.bormann-6lowpan-roadmap]). To enable the use of GHC towards a neighbor, a 6LoWPAN node needs to know that the neighbor implements it. While this can also simply be administratively required, a transition strategy as well as a way to support mixed networks is required.

One way to know a neighbor does implement GHC is receiving a packet from that neighbor with GHC in it ("implicit capability detection"). However, there needs to be a way to bootstrap this, as nobody ever would start sending packets with GHC otherwise.

To minimize the impact on [RFC6775], we define an ND option 6LoWPAN Capability Indication (6CIO), as illustrated in Figure 5.

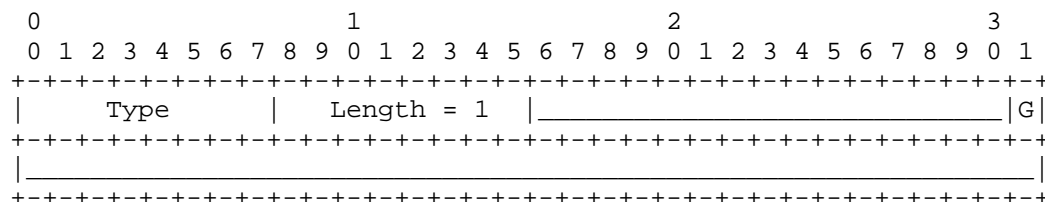


Figure 5: 6LoWPAN Capability Indication Option (6CIO)

The G bit indicates whether the node sending the option is GHC capable.

Once a node receives either an explicit or an implicit indication of GHC capability from another node, it may send GHC-compressed packets to that node. Where that capability has not been recently confirmed,

similar to the way PLPMTUD [RFC4821] finds out about changes in the network, a node SHOULD make use of NUD (neighbor unreachability detection) failures to switch back to basic 6LoWPAN header compression [RFC6282].

### 3.4. Using the 6CIO Option

The 6CIO option will typically only be ever sent in 6LoWPAN-ND RS packets (which cannot itself be GHC compressed unless the host desires to limit itself to talking to GHC capable routers). The resulting 6LoWPAN-ND RA can then already make use of GHC and thus indicate GHC capability implicitly, which in turn allows both nodes to use GHC in the 6LoWPAN-ND NS/NA exchange.

6CIO can also be used for future options that need to be negotiated between 6LoWPAN peers; an IANA registry is used to assign the flags. Bits marked by underscores in Figure 5 are unassigned and available for future assignment. They MUST be sent as zero and MUST be ignored on reception until assigned by IANA. Length values larger than 1 MUST be accepted by implementations in order to enable future extensions; the additional bits in the option are then deemed unassigned in the same way. For the purposes of the IANA registry, the bits are numbered in most-significant-bit-first order from the 16th bit of the option onward, i.e., the G bit is flag number 15. (Additional bits may also be used by a follow-on version of this document if some bit combinations that have been left unassigned here are then used in an upward compatible manner.)

Where the use of this option by other specifications is envisioned, the following items have to be kept in mind:

- o The option can be used in any ND packet.
- o Specific bits are set in the option to indicate that a capability is present in the sender. (There may be other ways to infer this information, as is the case in this specification.) Bit combinations may be used as desired. The absence of the capability `_indication_` is signaled by setting these bits to zero; this does not necessarily mean that the capability is absent.
- o The intention is not to modify the semantics of the specific ND packet carrying the option, but to provide the general capability indication described above.
- o Specifications have to be designed such that receivers that do not receive or do not process such a capability indication can still interoperate (presumably without exploiting the indicated capability).

- o The option is meant to be used sparsely, i.e. once a sender has reason to believe the capability indication has been received, there no longer is a need to continue sending it.

#### 4. IANA considerations

[This section to be removed/replaced by the RFC Editor.]

In the IANA registry for the "LOWPAN\_NHC Header Type" (in the "IPv6 Low Power Personal Area Network Parameters"), IANA needs to add the assignments in Figure 6.

10110IIN: Extension header GHC	[RFCthis]
11010CPP: UDP GHC	[RFCthis]
11011111: ICMPv6 GHC	[RFCthis]

Figure 6: IANA assignments for the NHC byte

IANA needs to allocate an ND option number for the 6CIO ND option format in the Registry "IPv6 Neighbor Discovery Option Formats" [RFC4861].

IANA needs to create a registry for "6LoWPAN capability bits" within the "Internet Control Message Protocol version 6 (ICMPv6) Parameters". The bits are assigned by giving their numbers as small non-negative integers as defined in section Section 3.4, preferably in the range 0..47. The policy is "RFC Required" [RFC5226]. The initial content of the registry is as in Figure 7:

0..14: unassigned	
15: GHC capable bit (G bit)	[RFCthis]
16..47: unassigned	

Figure 7



## 5. Security considerations

The security considerations of [RFC4944] and [RFC6282] apply. As usual in protocols with packet parsing/construction, care must be taken in implementations to avoid buffer overflows and in particular (with respect to the back-referencing) out-of-area references during decompression.

One additional consideration is that an attacker may send a forged packet that makes a second node believe a third victim node is GHC-capable. If it is not, this may prevent packets sent by the second node from reaching the third node (at least until robustness features such as those discussed in Section 3.3 kick in).

No mitigation is proposed (or known) for this attack, except that a victim node that does implement GHC is not vulnerable. However, with unsecured ND, a number of attacks with similar outcomes are already possible, so there is little incentive to make use of this additional attack. With secured ND, 6CIO is also secured; nodes relying on secured ND therefore should use 6CIO bidirectionally (and limit the implicit capability detection to secured ND packets carrying GHC) instead of basing their neighbor capability assumptions on receiving any kind of unprotected packet.

## 6. Acknowledgements

Colin O'Flynn has repeatedly insisted that some form of compression for ICMPv6 and ND packets might be beneficial. He actually wrote his own draft, [I-D.oflynn-6lowpan-icmphc], which compresses better, but addresses basic ICMPv6/ND only and needs a much longer spec (around 17 pages of detailed spec, as compared to the single page of core spec here). This motivated the author to try something simple, yet general. Special thanks go to Colin for indicating that he indeed considers his draft superseded by the present one.

The examples given are based on pcap files that Colin O'Flynn, Owen Kirby, Olaf Bergmann and others provided.

The static dictionary was developed, and the bit allocations validated, based on research by Sebastian Dominik.

Erik Nordmark provided input that helped shaping the 6CIO option.

Yoshihiro Ohba insisted on clarifying the notation used for the definition of the bytetimes and their bitfields.

## 7. References

### 7.1. Normative References

- [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.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [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.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

### 7.2. Informative References

- [I-D.bormann-6lowpan-roadmap] Bormann, C., "6LoWPAN Roadmap and Implementation Guide", draft-bormann-6lowpan-roadmap-04 (work in progress), April 2013.
- [I-D.oflynn-6lowpan-icmphc] O'Flynn, C., "ICMPv6/ND Compression for 6LoWPAN Networks", draft-oflynn-6lowpan-icmphc-00 (work in progress), July 2010.
- [RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.
- [RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le,

K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, March 2010.

## Appendix A. Examples

This section demonstrates some relatively realistic examples derived from actual PCAP dumps taken at previous interops.

Figure 8 shows an RPL DODAG Information Solicitation, a quite short RPL message that obviously cannot be improved much.

IP header:

```
60 00 00 00 00 08 3a ff fe 80 00 00 00 00 00 00
02 1c da ff fe 00 20 24 ff 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 1a
```

Payload:

```
9b 00 6b de 00 00 00 00
```

Dictionary:

```
fe 80 00 00 00 00 00 00 00 02 1c da ff fe 00 20 24
ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
00 00 00 08 00 00 00 00 3a 16 fe fd 17 fe fd 00 01
00 00 00 00 00 00 01 00 00
```

copy: 04 9b 00 6b de

4 nulls: 82

Compressed:

```
04 9b 00 6b de 82
```

Was 8 bytes; compressed to 6 bytes, compression factor 1.33

Figure 8: A simple RPL example

Figure 9 shows an RPL DODAG Information Object, a longer RPL control message that is improved a bit more. Note that the compressed output exposes an inefficiency in the simple-minded compressor used to generate it; this does not devalue the example since constrained nodes are quite likely to make use of simple-minded compressors.

```

IP header:
 60 00 00 00 00 5c 3a ff fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23 ff 02 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 1a
Payload:
 9b 01 7a 5f 00 f0 01 00 88 00 00 00 20 02 0d b8
 00 00 00 00 00 00 00 00 ff fe 00 fa ce 04 0e 00 14
 09 ff 00 00 01 00 00 00 00 00 00 00 08 1e 80 20
 ff ff ff ff ff ff ff ff 00 00 00 00 20 02 0d b8
 00 00 00 00 00 00 00 00 ff fe 00 fa ce 03 0e 40 00
 ff ff ff ff 20 02 0d b8 00 00 00 00
Dictionary:
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
 00 00 00 5c 00 00 00 3a 16 fe fd 17 fe fd 00 01
 00 00 00 00 00 01 00 00
copy: 06 9b 01 7a 5f 00 f0
ref(9): 01 00 -> ref 1lnnnk 0 7: c7
copy: 01 88
3 nulls: 81
copy: 04 20 02 0d b8
7 nulls: 85
ref(68): ff fe 00 -> ref 10lnssss 0 8/1lnnnk 1 1: a8 c9
copy: 08 fa ce 04 0e 00 14 09 ff
ref(39): 00 00 01 00 00 -> ref 10lnssss 0 4/1lnnnk 3 2: a4 da
5 nulls: 83
copy: 06 08 1e 80 20 ff ff
ref(2): ff ff -> ref 1lnnnk 0 0: c0
ref(4): ff ff ff ff -> ref 1lnnnk 2 0: d0
4 nulls: 82
ref(48): 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 fa ce
-> ref 10lnssss 1 4/1lnnnk 6 0: b4 f0
copy: 03 03 0e 40
ref(9): 00 ff -> ref 1lnnnk 0 7: c7
ref(28): ff ff ff -> ref 10lnssss 0 3/1lnnnk 1 1: a3 c9
ref(24): 20 02 0d b8 00 00 00 00
-> ref 10lnssss 0 2/1lnnnk 6 0: a2 f0
Compressed:
 06 9b 01 7a 5f 00 f0 c7 01 88 81 04 20 02 0d b8
 85 a8 c9 08 fa ce 04 0e 00 14 09 ff a4 da 83 06
 08 1e 80 20 ff ff c0 d0 82 b4 f0 03 03 0e 40 c7
 a3 c9 a2 f0
Was 92 bytes; compressed to 52 bytes, compression factor 1.77

```

Figure 9: A longer RPL example

Similarly, Figure 10 shows an RPL DAO message. One of the embedded addresses is copied right out of the pseudo-header, the other one is effectively converted from global to local by providing the prefix FE80 literally, inserting a number of nulls, and copying (some of) the IID part again out of the pseudo-header. Note that a simple implementation would probably emit fewer nulls and copy the entire IID; there are multiple ways to encode this 50-byte payload into 27 bytes.

IP header:

```
60 00 00 00 00 32 3a ff 20 02 0d b8 00 00 00 00
00 00 00 ff fe 00 33 44 20 02 0d b8 00 00 00 00
00 00 00 ff fe 00 11 22
```

Payload:

```
9b 02 58 7d 01 80 00 f1 05 12 00 80 20 02 0d b8
00 00 00 00 00 00 00 ff fe 00 33 44 06 14 00 80
f1 00 fe 80 00 00 00 00 00 00 00 00 ff fe 00
11 22
```

Dictionary:

```
20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 33 44
20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 11 22
00 00 00 32 00 00 00 3a 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

copy: 0c 9b 02 58 7d 01 80 00 f1 05 12 00 80

ref(68): 20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 33 44

-> ref 10lnssss 1 6/1lnnnkkk 6 4: b6 f4

copy: 08 06 14 00 80 f1 00 fe 80

9 nulls: 87

ref(74): ff fe 00 11 22 -> ref 10lnssss 0 8/1lnnnkkk 3 5: a8 dd

Compressed:

```
0c 9b 02 58 7d 01 80 00 f1 05 12 00 80 b6 f4 08
06 14 00 80 f1 00 fe 80 87 a8 dd
```

Was 50 bytes; compressed to 27 bytes, compression factor 1.85

Figure 10: An RPL DAO message

Figure 11 shows the effect of compressing a simple ND neighbor solicitation.

IP header:

```
60 00 00 00 00 30 3a ff 20 02 0d b8 00 00 00 00
00 00 00 ff fe 00 3b d3 fe 80 00 00 00 00 00 00
02 1c da ff fe 00 30 23
```

Payload:

```
87 00 a7 68 00 00 00 00 fe 80 00 00 00 00 00 00
02 1c da ff fe 00 30 23 01 01 3b d3 00 00 00 00
1f 02 00 00 00 00 00 06 00 1c da ff fe 00 20 24
```

Dictionary:

```
20 02 0d b8 00 00 00 00 00 00 00 ff fe 00 3b d3
fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
00 00 00 30 00 00 00 3a 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

copy: 04 87 00 a7 68

4 nulls: 82

ref(48): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23

-> ref 10lnssss 1 4/1lnnnkkk 6 0: b4 f0

copy: 04 01 01 3b d3

4 nulls: 82

copy: 02 1f 02

5 nulls: 83

copy: 02 06 00

ref(24): 1c da ff fe 00 -> ref 10lnssss 0 2/1lnnnkkk 3 3: a2 db

copy: 02 20 24

Compressed:

```
04 87 00 a7 68 82 b4 f0 04 01 01 3b d3 82 02 1f
02 83 02 06 00 a2 db 02 20 24
```

Was 48 bytes; compressed to 26 bytes, compression factor 1.85

Figure 11: An ND neighbor solicitation

Figure 12 shows the compression of an ND neighbor advertisement.

IP header:

```
60 00 00 00 00 30 3a fe fe 80 00 00 00 00 00 00
02 1c da ff fe 00 30 23 20 02 0d b8 00 00 00 00
00 00 00 ff fe 00 3b d3
```

Payload:

```
88 00 26 6c c0 00 00 00 fe 80 00 00 00 00 00 00
02 1c da ff fe 00 30 23 02 01 fa ce 00 00 00 00
1f 02 00 00 00 00 00 06 00 1c da ff fe 00 20 24
```

Dictionary:

```
fe 80 00 00 00 00 00 02 1c da ff fe 00 30 23
20 02 0d b8 00 00 00 00 00 00 ff fe 00 3b d3
00 00 00 30 00 00 00 3a 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

copy: 05 88 00 26 6c c0

3 nulls: 81

ref(64): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23

-> ref 10lnssss 1 6/1lnnnkkk 6 0: b6 f0

copy: 04 02 01 fa ce

4 nulls: 82

copy: 02 1f 02

5 nulls: 83

copy: 02 06 00

ref(24): 1c da ff fe 00 -> ref 10lnssss 0 2/1lnnnkkk 3 3: a2 db

copy: 02 20 24

Compressed:

```
05 88 00 26 6c c0 81 b6 f0 04 02 01 fa ce 82 02
1f 02 83 02 06 00 a2 db 02 20 24
```

Was 48 bytes; compressed to 27 bytes, compression factor 1.78

Figure 12: An ND neighbor advertisement

Figure 13 shows the compression of an ND router solicitation. Note that the relatively good compression is not caused by the many zero bytes in the link-layer address of this particular capture (which are unlikely to occur in practice): 7 of these 8 bytes are copied from the pseudo-header (the 8th byte cannot be copied as the universal/local bit needs to be inverted).

IP header:

```
60 00 00 00 00 18 3a ff fe 80 00 00 00 00 00 00
ae de 48 00 00 00 00 01 ff 02 00 00 00 00 00 00
00 00 00 00 00 00 00 02
```

Payload:

```
85 00 90 65 00 00 00 00 01 02 ac de 48 00 00 00
00 01 00 00 00 00 00 00
```

Dictionary:

```
fe 80 00 00 00 00 00 00 ae de 48 00 00 00 00 01
ff 02 00 00 00 00 00 00 00 00 00 00 00 00 02
00 00 00 18 00 00 00 3a 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

copy: 04 85 00 90 65

ref(11): 00 00 00 00 01 -> ref 11nnnk 3 6: de

copy: 02 02 ac

ref(58): de 48 00 00 00 00 01

-> ref 10lnssss 0 6/11nnnk 5 3: a6 eb

6 nulls: 84

Compressed:

```
04 85 00 90 65 de 02 02 ac a6 eb 84
```

Was 24 bytes; compressed to 12 bytes, compression factor 2.00

Figure 13: An ND router solicitation

Figure 14 shows the compression of an ND router advertisement. The indefinite lifetime is compressed to four bytes by backreferencing; this could be improved (at the cost of minor additional decompressor complexity) by including some simple runlength mechanism.



```

IP header:
 60 00 00 00 00 60 3a ff fe 80 00 00 00 00 00 00
 10 34 00 ff fe 00 11 22 fe 80 00 00 00 00 00 00
 ae de 48 00 00 00 00 01
Payload:
 86 00 55 c9 40 00 0f a0 1c 5a 38 17 00 00 07 d0
 01 01 11 22 00 00 00 00 03 04 40 40 ff ff ff ff
 ff ff ff ff 00 00 00 00 20 02 0d b8 00 00 00 00
 00 00 00 00 00 00 00 00 20 02 40 10 00 00 03 e8
 20 02 0d b8 00 00 00 00 21 03 00 01 00 00 00 00
 20 02 0d b8 00 00 00 00 00 00 00 00 ff fe 00 11 22
Dictionary:
 fe 80 00 00 00 00 00 00 10 34 00 ff fe 00 11 22
 fe 80 00 00 00 00 00 00 ae de 48 00 00 00 00 01
 00 00 00 60 00 00 00 3a 16 fe fd 17 fe fd 00 01
 00 00 00 00 00 01 00 00
copy: 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17
2 nulls: 80
copy: 06 07 d0 01 01 11 22
4 nulls: 82
copy: 06 03 04 40 40 ff ff
ref(2): ff ff -> ref 1lnnnk 0 0: c0
ref(4): ff ff ff ff -> ref 1lnnnk 2 0: d0
4 nulls: 82
copy: 04 20 02 0d b8
12 nulls: 8a
copy: 04 20 02 40 10
ref(38): 00 00 03 -> ref 10lnss 0 4/1lnnnk 1 3: a4 cb
copy: 01 e8
ref(24): 20 02 0d b8 00 00 00 00
-> ref 10lnss 0 2/1lnnnk 6 0: a2 f0
copy: 02 21 03
ref(84): 00 01 00 00 00 00
-> ref 10lnss 0 9/1lnnnk 4 6: a9 e6
ref(40): 20 02 0d b8 00 00 00 00 00 00 00
-> ref 10lnss 1 3/1lnnnk 1 5: b3 cd
ref(136): ff fe 00 11 22
-> ref 10lnss 0 15/10lnss 0 1/1lnnnk 3 3: af a1 db
Compressed:
 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17 80 06 07
 d0 01 01 11 22 82 06 03 04 40 40 ff ff c0 d0 82
 04 20 02 0d b8 8a 04 20 02 40 10 a4 cb 01 e8 a2
 f0 02 21 03 a9 e6 b3 cd af a1 db
Was 96 bytes; compressed to 59 bytes, compression factor 1.63

```

Figure 14: An ND router advertisement

Figure 15 shows the compression of a DTLS application data packet with a net payload of 13 bytes of cleartext, and 8 bytes of authenticator (note that the IP header is not relevant for this example and has been set to 0). This makes good use of the static dictionary, and is quite effective crunching out the redundancy in the TLS\_PSK\_WITH\_AES\_128\_CCM\_8 header, leading to a net reduction by 15 bytes.

IP header:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

Payload:

```
17 fe fd 00 01 00 00 00 00 00 01 00 1d 00 01 00
00 00 00 00 01 09 b2 0e 82 c1 6e b6 96 c5 1f 36
8d 17 61 e2 b5 d4 22 d4 ed 2b
```

Dictionary:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 2a 00 00 00 00 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

ref(13): 17 fe fd 00 01 00 00 00 00 00 00 01 00

-> ref 10lnssss 1 0/1lnnnkkk 2 1: b0 d1

copy: 01 1d

ref(10): 00 01 00 00 00 00 00 00 01 -> ref 1lnnnkkk 6 2: f2

copy: 15 09 b2 0e 82 c1 6e b6 96 c5 1f 36 8d 17 61 e2

copy: b5 d4 22 d4 ed 2b

Compressed:

```
b0 d1 01 1d f2 15 09 b2 0e 82 c1 6e b6 96 c5 1f
36 8d 17 61 e2 b5 d4 22 d4 ed 2b
```

Was 42 bytes; compressed to 27 bytes, compression factor 1.56

Figure 15: A DTLS application data packet

Figure 16 shows that the compression is slightly worse in a subsequent packet (containing 6 bytes of cleartext and 8 bytes of authenticator, yielding a net compression of 13 bytes). The total overhead does stay at a quite acceptable 8 bytes.

IP header:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

Payload:

```
17 fe fd 00 01 00 00 00 00 05 00 16 00 01 00
00 00 00 00 05 ae a0 15 56 67 92 4d ff 8a 24 e4
cb 35 b9
```

Dictionary:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 23 00 00 00 00 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

ref(13): 17 fe fd 00 01 00 00 00 00 00

-> ref 10lnssss 1 0/1lnnnkkk 0 3: b0 c3

copy: 03 05 00 16

ref(10): 00 01 00 00 00 00 00 05 -> ref 1lnnnkkk 6 2: f2

copy: 0e ae a0 15 56 67 92 4d ff 8a 24 e4 cb 35 b9

Compressed:

```
b0 c3 03 05 00 16 f2 0e ae a0 15 56 67 92 4d ff
8a 24 e4 cb 35 b9
```

Was 35 bytes; compressed to 22 bytes, compression factor 1.59

Figure 16: Another DTLS application data packet

Figure 17 shows the compression of a DTLS handshake message, here a client hello. There is little that can be compressed about the 32 bytes of randomness. Still, the net reduction is by 14 bytes.

IP header:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

Payload:

```
16 fe fd 00 00 00 00 00 00 00 00 00 00 36 01 00 00
2a 00 00 00 00 00 00 00 2a fe fd 51 52 ed 79 a4
20 c9 62 56 11 47 c9 39 ee 6c c0 a4 fe c6 89 2f
32 26 9a 16 4e 31 7e 9f 20 92 92 00 00 00 02 c0
a8 01 00
```

Dictionary:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 43 00 00 00 00 16 fe fd 17 fe fd 00 01
00 00 00 00 00 01 00 00
```

ref(16): 16 fe fd -> ref 10lnsssss 0 1/1lnnnkkkk 1 5: a1 cd

9 nulls: 87

copy: 01 36

ref(16): 01 00 00 -> ref 10lnsssss 0 1/1lnnnkkkk 1 5: a1 cd

copy: 01 2a

7 nulls: 85

copy: 23 2a fe fd 51 52 ed 79 a4 20 c9 62 56 11 47 c9

copy: 39 ee 6c c0 a4 fe c6 89 2f 32 26 9a 16 4e 31 7e

copy: 9f 20 92 92

3 nulls: 81

copy: 05 02 c0 a8 01 00

Compressed:

```
a1 cd 87 01 36 a1 cd 01 2a 85 23 2a fe fd 51 52
ed 79 a4 20 c9 62 56 11 47 c9 39 ee 6c c0 a4 fe
c6 89 2f 32 26 9a 16 4e 31 7e 9f 20 92 92 81 05
02 c0 a8 01 00
```

Was 67 bytes; compressed to 53 bytes, compression factor 1.26

Figure 17: A DTLS handshake packet (client hello)

Author's Address

Carsten Bormann  
Universitaet Bremen TZI  
Postfach 330440  
D-28359 Bremen  
Germany

Phone: +49-421-218-63921

Email: [cabo@tzi.org](mailto:cabo@tzi.org)

IPv6 Maintenance WG  
Internet-Draft  
Intended status: Standards Track  
Expires: December 20, 2013

A. Brandt  
J. Buron  
Sigma Designs  
June 18, 2013

Transmission of IPv6 packets over ITU-T G.9959 Networks  
draft-brandt-6man-lowpanz-02

## Abstract

This document describes the frame format for transmission of IPv6 packets and a method of forming IPv6 link-local addresses and statelessly autoconfigured IPv6 addresses on ITU-T G.9959 networks.

## Requirements Language

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

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 20, 2013.

## Copyright Notice

Copyright (c) 2013 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. Author's notes . . . . .	2
1.1. Reader's guidance . . . . .	2
2. Introduction . . . . .	3
2.1. Terms used . . . . .	3
3. G.9959 parameters to use for IPv6 transport . . . . .	4
3.1. Addressing mode . . . . .	4
3.2. IPv6 Multicast support . . . . .	4
3.3. G.9959 MAC PDU size and IPv6 MTU . . . . .	5
3.4. Transmission status indications . . . . .	5
3.5. Transmission security . . . . .	5
4. LoWPAN Adaptation Layer and Frame Format . . . . .	6
4.1. Dispatch Header . . . . .	6
5. LoWPAN addressing . . . . .	7
5.1. Stateless Address Autoconfiguration of routable IPv6 addresses . . . . .	8
5.2. IPv6 Link Local Address . . . . .	8
5.3. Unicast Address Mapping . . . . .	8
5.4. On the use of Neighbor Discovery technologies . . . . .	9
5.4.1. Prefix and CID management (Route-over) . . . . .	10
5.4.2. Prefix and CID management (Mesh-under) . . . . .	10
6. Header Compression . . . . .	10
7. IANA Considerations . . . . .	11
8. Security Considerations . . . . .	11
9. Acknowledgements . . . . .	12
10. References . . . . .	12
10.1. Normative References . . . . .	12
10.2. Informative References . . . . .	13
Authors' Addresses . . . . .	14

## 1. Author's notes

This chapter MUST be deleted before going for document last call.

### 1.1. Reader's guidance

This document borrows heavily from RFC4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks". The process of creating this document was mainly a simplification; removing the following topics:

- o EUI-64 link-layer addresses

- o Fragmentation layer
- o Mesh routing

The 16-bit short addresses of 802.15.4 have been changed to 8-bit G.9959 NodeIDs.

## 2. Introduction

The ITU-T G.9959 recommendation [G.9959] targets low-power Personal Area Networks (PANs). This document defines the frame format for transmission of IPv6 [RFC2460] packets as well as the formation of IPv6 link-local addresses and statelessly autoconfigured IPv6 addresses on G.9959 networks.

The general approach is to adapt elements of [RFC4944] to G.9959 networks. G.9959 provides a Segmentation and Reassembly (SAR) layer for transmission of datagrams larger than the G.9959 MAC PDU.

[RFC6775] updates [RFC4944] by specifying 6LoWPAN optimizations for IPv6 Neighbor Discovery (ND) (originally defined by [RFC4861]). This document limits the use of [RFC6775] to prefix and Context ID assignment. It is described how to construct an IID from a G.9959 link-layer address. Refer to Section 5. If using that method, Duplicate Address Detection (DAD) is not needed. Address registration is only needed in certain cases.

In addition to IPv6 application communication, the frame format defined in this document may be used by IPv6 routing protocols such as RPL [RFC6550] or P2P-RPL [P2P-RPL] to implement IPv6 routing over G.9959 networks.

G.9959 networks may implement mesh routing between nodes below the IP layer. Mesh routing is out of scope of this document.

### 2.1. Terms used

ABR: Authoritative Border Router ([RFC6775])

AES: Advanced Encryption Scheme

EUI-64: Extended Unique Identifier

HomeID: G.9959 Link-Layer Network Identifier

IID: Interface IDentifier

MAC: Media Access Control



MTU: Maximum Transmission Unit

NodeID: G.9959 Link-Layer Node Identifier (Short Address)

PAN: Personal Area Network

PDU: Protocol Data Unit

SAR: Segmentation And Reassembly

ULA: Unique Local Address

### 3. G.9959 parameters to use for IPv6 transport

This chapter outlines properties applying to the PHY and MAC of G.9959 and how to use these for IPv6 transport.

#### 3.1. Addressing mode

G.9959 defines how a unique 32-bit HomeID network identifier is assigned by a network controller and how an 8-bit NodeID host identifier is allocated. NodeIDs are unique within the logical network identified by the HomeID. The logical network identified by the HomeID maps directly to an IPv6 subnet identified by one or more IPv6 prefixes.

An IPv6 host SHOULD construct its link-local IPv6 address and routable IPv6 addresses from the NodeID in order to facilitate IP header compression as described in [RFC6282].

A word of caution: since HomeIDs and NodeIDs are handed out by a network controller function during inclusion, identifier validity and uniqueness is limited by the lifetime of the logical network membership. This can be cut short by a mishap occurring to the network controller. Having a single point of failure at the network controller suggests that deployers of high-reliability applications should carefully consider adding redundancy to the network controller function.

#### 3.2. IPv6 Multicast support

[RFC3819] recommends that IP subnetworks support (subnet-wide) multicast. G.9959 supports direct-range IPv6 multicast while subnet-wide multicast is not supported natively by G.9959. Subnet-wide multicast may be provided by an IP routing protocol or a mesh routing protocol operating below the 6LoWPAN layer. Routing protocols are out of scope of this document.

IPv6 multicast packets MUST be carried via G.9959 broadcast.

As per [G.9959], this is accomplished as follows:

1. The destination HomeID of the G.9959 MAC PDU MUST be the HomeID of the logical network
2. The destination NodeID of the G.9959 MAC PDU MUST be the broadcast NodeID (0xff)

G.9959 broadcast MAC PDUs are only intercepted by nodes within the logical network identified by the HomeID.

### 3.3. G.9959 MAC PDU size and IPv6 MTU

IPv6 packets MUST use G.9959 transmission profiles which support MAC PDU payload sizes of 150 bytes or higher, e.g. the R3 profile. G.9959 profiles R1 and R2 only supports MPDU payloads around 40 bytes and the transmission speed is down to 9.6kbit/s.

[RFC2460] specifies that IPv6 packets may be up to 1280 octets. However, a full IPv6 packet does not fit in an G.9959 MAC PDU. The maximum G.9959 R3 MAC PDU payload size is 158 octets. Link-layer security imposes an overhead, which in the extreme case leaves 130 octets available.

G.9959 provides Segmentation And Reassembly for payloads up to 1350 octets. Segmentation however adds further overhead. It is therefore desirable that datagrams can fit into a single G.9959 MAC PDU. IPv6 Header Compression [RFC6282] improves the chances that a short IPv6 packet can fit into a single G.9959 frame.

### 3.4. Transmission status indications

The G.9959 MAC layer provides native acknowledgement and retransmission of MAC PDUs. The G.9959 SAR layer does the same for larger datagrams. A mesh routing layer may provide a similar feature for routed communication. Acknowledgment and retransmission improves the transmission success rate and frees higher layers from the burden of implementing individual retransmission schemes. An IPv6 routing stack communicating over G.9959 may utilize link-layer status indications such as delivery confirmation and Ack timeout from the MAC layer.

### 3.5. Transmission security

Implementations claiming conformance with this document MUST enable G.9959 shared network key security.

The shared network key is intended to address security requirements in the home at the normal security requirements level. For applications with high or very high requirements on confidentiality and/or integrity, additional application layer security measures for end-to-end authentication and encryption may need to be applied. The availability of the network relies on the security properties of the network key in any case.

#### 4. LoWPAN Adaptation Layer and Frame Format

The 6LoWPAN encapsulation formats defined in this chapter are the payload in the G.9959 MAC PDU. IPv6 header compression [RFC6282] MUST be supported by implementations of this specification.

All 6LoWPAN datagrams transported over G.9959 are prefixed by a 6LoWPAN encapsulation header stack. The 6LoWPAN payload (e.g. an IPv6 packet) follows this encapsulation header. Each header in the header stack contains a header type followed by zero or more header fields. An IPv6 header stack may contain, in the following order, addressing, hop-by-hop options, routing, fragmentation, destination options, and finally payload [RFC2460]. The 6LoWPAN header format is structured the same way. Currently only payload options are defined for the 6LoWPAN header format.

The definition of 6LoWPAN headers consists of the dispatch value, the definition of the header fields that follow, and their ordering constraints relative to all other headers. Although the header stack structure provides a mechanism to address future demands on the 6LoWPAN adaptation layer, it is not intended to provide general purpose extensibility. This document specifies a small set of 6LoWPAN header types using the 6LoWPAN header stack for clarity, compactness, and orthogonality.

#### 4.1. Dispatch Header

The dispatch header is shown below:

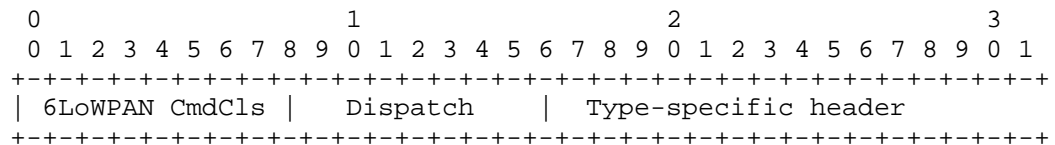


Figure 1: Dispatch Type and Header

6LoWPAN CmdCls: 6LoWPAN Command Class identifier. This field MUST carry the value 0x4F [G.9959]. The value specifies that the

following bits are a 6LoWPAN encapsulated datagram. Non-6LoWPAN protocols MUST ignore the contents following the 6LoWPAN Command Class identifier.

Dispatch: Identifies the header type immediately following the Dispatch Header.

Type-specific header: A header determined by the Dispatch Header.

The dispatch value may be treated as an unstructured namespace. Only a few symbols are required to represent current 6LoWPAN functionality. Although some additional savings could be achieved by encoding additional functionality into the dispatch byte, these measures would tend to constrain the ability to address future alternatives.

Dispatch values used in this specification are compatible with the dispatch values defined by [RFC4944] and [RFC6282].

Pattern	Header Type	Reference
01 000001	IPv6 - Uncompressed IPv6 Addresses	[RFC4944]
01 1xxxxx	6LoWPAN_IPHC - 6LoWPAN_IPHC compressed IPv6	[RFC6282]

All other Dispatch values are unassigned in this document.

Figure 2: Dispatch values

IPv6: Specifies that the following header is an uncompressed IPv6 header.

6LoWPAN\_IPHC: IPv6 Header Compression. Refer to [RFC6282].

## 5. LoWPAN addressing

IPv6 addresses are autoconfigured from IIDs which are again constructed from link-layer address information to save memory in devices and to facilitate efficient IP header compression as per [RFC6282].

A G.9959 NodeID is 8 bits in length. A NodeID is mapped into an IEEE EUI-64 identifier as follows:

`IID = 0000:00ff:fe00:YXXX`

Figure 3: Constructing a compressible IID

where XX carries the G.9959 NodeID and YY is a one byte value chosen by the individual node. The default YY value MUST be zero. A node MAY use other values of YY than zero to form additional IIDs in order to instantiate multiple IPv6 interfaces. The YY value MUST be ignored when computing the corresponding NodeID (the XX value) from an IID.

A 6LoWPAN network typically is used for M2M-style communication. The method of constructing IIDs from the link-layer address obviously does not support addresses assigned or constructed by other means. A node MUST NOT compute the NodeID from the IID if the first 6 bytes of the IID do not comply with the format defined in Figure 3. In that case, the address resolution mechanisms of RFC 6775 apply.

#### 5.1. Stateless Address Autoconfiguration of routable IPv6 addresses

The IID defined above MUST be used whether autoconfiguring a ULA IPv6 address [RFC4193] or a globally routable IPv6 address [RFC3587] in G.9959 subnets.

#### 5.2. IPv6 Link Local Address

The IPv6 link-local address [RFC4291] for a G.9959 interface is formed by appending the IID defined above to the IPv6 link local prefix FE80::/64.

The "Universal/Local" (U/L) bit MUST be set to zero in keeping with the fact that this is not a globally unique value [EUI64].

The resulting link local address is formed as follows:

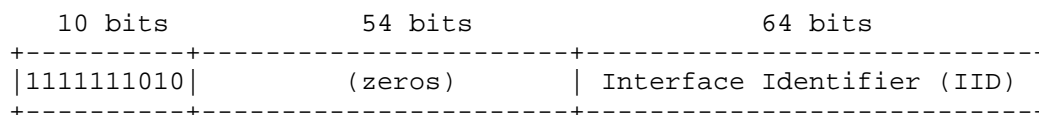


Figure 4: IPv6 Link Local Address

#### 5.3. Unicast Address Mapping

The address resolution procedure for mapping IPv6 unicast addresses into G.9959 link-layer addresses follows the general description in

Section 7.2 of [RFC4861]. The Source/Target Link-layer Address option MUST have the following form when the link layer is G.9959.

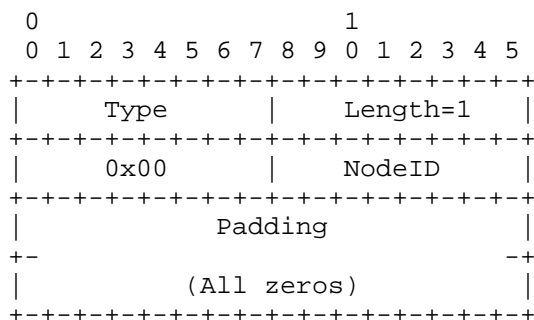


Figure 5: IPv6 Unicast Address Mapping

Option fields:

Type: The value 1 signifies the Source Link-layer address. The value 2 signifies the Destination Link-layer address.

Length: This is the length of this option (including the type and length fields) in units of 8 octets. The value of this field is always 1 for G.9959 NodeIDs.

NodeID: This is the G.9959 NodeID the actual interface currently responds to. The link-layer address may change if the interface joins another network at a later time.

#### 5.4. On the use of Neighbor Discovery technologies

[RFC4861] specifies how IPv6 nodes may resolve link layer addresses from IPv6 addresses via the use of link-local IPv6 multicast. [RFC6775] is an optimization of [RFC4861], specifically targeting 6LoWPAN networks. [RFC6775] defines how a 6LoWPAN node may register IPv6 addresses with an authoritative border router (ABR). Generally, nodes SHOULD NOT use [RFC6775] address registration. However, address registration MUST be used if the first 6 bytes of the IID do not comply with the format defined in Figure 3.

In route-over environments, IPv6 hosts MUST use [RFC6775] address registration. [RFC6775] Duplicate Address Detection (DAD) SHOULD NOT be used, since the link-layer inclusion process of G.9959 ensures that a NodeID is unique for a given HomeID.

#### 5.4.1. Prefix and CID management (Route-over)

A node implementation for route-over operation MAY use RFC6775 mechanisms for obtaining IPv6 prefixes and corresponding header compression context information [RFC6282]. RFC6775 Route-over requirements apply with no modifications.

#### 5.4.2. Prefix and CID management (Mesh-under)

An implementation for mesh-under operation MUST use [RFC6775] mechanisms for managing IPv6 prefixes and corresponding header compression context information [RFC6282]. When using [RFC6775] mechanisms for sending RAs, the M flag MUST NOT be set. As stated by [RFC6775], an ABR is responsible for managing prefix(es). Global prefixes may change over time. It is RECOMMENDED that a ULA prefix is always assigned to the 6LoWPAN subnet to facilitate stable site-local application associations based on IPv6 addresses. Prefixes used in the 6LoWPAN subnet are distributed by normal RA mechanisms. The 6LoWPAN Context Option (6CO) is used according to [RFC6775] in an RA to disseminate Context IDs (CID) to use for compressing prefixes. Prefixes and corresponding Context IDs MUST be assigned during initial node inclusion. Nodes MUST renew the prefix and CID according to the lifetime signaled by the ABR. [RFC6775] specifies that the maximum value of the RA Router Lifetime field MAY be up to 0xFFFF. This document further specifies that the value 0xFFFF MUST be interpreted as infinite lifetime. This value SHOULD NOT be used by ABRs. Its use is only intended for a sleeping network controller; for instance a battery powered remote control being master for a small island-mode network of light modules. CIDs SHOULD be used in a cyclic fashion to assist battery powered nodes with no real-time clock. When updating context information, a CID may have its lifetime set to zero to obsolete it. The CID SHOULD NOT be reused immediately; rather the next vacant CID should be assigned. An ABR detecting the use of an obsoleted CID SHOULD immediately send an RA with updated Context Information. Header compression based on CIDs MUST NOT be used for RA messages carrying Context Information. An expired CID and the associated prefix SHOULD NOT be reset but rather retained in receive-only mode if there is no other current need for the CID value. This will allow an ABR to detect if a sleeping node without clock uses an expired CID and in response, the LBR SHOULD immediately return an RA with fresh Context Information to the originator. Except for the specific redefinition of the RA Router Lifetime value 0xFFFF, the above text is in compliance with [RFC6775].

## 6. Header Compression

IPv6 header fields SHOULD be compressed. If IPv6 header compression is used, it MUST be according to [RFC6282]. This section will simply identify substitutions that should be made when interpreting the text of [RFC6282].

In general the following substitutions should be made:

- o Replace "802.15.4" with "G.9959"
- o Replace "802.15.4 short address" with "<Interface><G.9959 NodeID>"
- o Replace "802.15.4 PAN ID" with "G.9959 HomeID"

When a 16-bit address is called for (i.e., an IEEE 802.15.4 "short address") it MUST be formed by prepending an Interface label byte to the G.9959 NodeID:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|   Interface   |   NodeID   |
+---+---+---+---+---+---+---+---+

```

A transmitting node may be sending to an IPv6 destination address which can be reconstructed from the link-layer destination address. If the Interface number is zero (the default value), all IPv6 address bytes may be elided. Likewise, the Interface number of a fully elided IPv6 address (i.e. SAM/DAM=11) may be reconstructed to the value zero by a receiving node.

64 bit 802.15.4 address details MUST be ignored. This document only specifies the use of short addresses.

## 7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

## 8. Security Considerations

The method of derivation of Interface Identifiers from 8-bit NodeIDs preserves uniqueness within the logical network. However, there is no protection from duplication through forgery. Neighbor Discovery in G.9959 links may be susceptible to threats as detailed in [RFC3756]. G.9959 networks may feature mesh routing. This implies



additional threats due to ad hoc routing as per [KW03]. G.9959 provides capability for link-layer security. G.9959 nodes MUST use link-layer security with a shared key. Doing so will alleviate the majority of threats stated above. A sizeable portion of G.9959 devices is expected to always communicate within their PAN (i.e., within their subnet, in IPv6 terms). In response to cost and power consumption considerations, these devices will typically implement the minimum set of features necessary. Accordingly, security for such devices may rely on the mechanisms defined at the link layer by G.9959. G.9959 relies on the Advanced Encryption Standard (AES) for authentication and encryption of G.9959 frames and further employs challenge-response handshaking to prevent replay attacks.

It is also expected that some G.9959 devices (e.g. billing and/or safety critical products) will implement coordination or integration functions. These may communicate regularly with IPv6 peers outside the subnet. Such IPv6 devices are expected to secure their end-to-end communications with standard security mechanisms (e.g., IPsec, TLS, etc).

## 9. Acknowledgements

Thanks to the authors of RFC 4944 and RFC 6282 and members of the IETF 6LoWPAN working group; this document borrows extensively from their work. Thanks to Kerry Lynn, Tommas Jess Christensen and Erez Ben-Tovim for useful discussions. Thanks to Carsten Bormann for extensive feedback which improved this document significantly.

## 10. References

### 10.1. Normative References

- [EUI64] IEEE, "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", IEEE Std <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>, November 2012.
- [G.9959.llc] ITU-T, "G.9959 Contribution: Logical Link Control (LLC) layer", ITU-T draft contribution 2013-04-Q15-023.doc, April 2013.
- [G.9959.sar] ITU-T, "G.9959 Contribution: Segmentation And Reassembly (SAR) adaptation layer", ITU-T draft contribution 2013-04-Q15-024.doc, April 2013.

- [G.9959] ITU-T, "G.9959: Low-Power, narrowband radio for control applications", January 2012.
- [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.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, August 2003.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

## 10.2. Informative References

- [P2P-RPL] Goyal, M., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "IETF, I-D.ietf-roll-p2p-rpl-15, Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks", December 2012.

- [RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.
- [RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

#### Authors' Addresses

Anders Brandt  
Sigma Designs  
Emdrupvej 26A, 1.  
Copenhagen O 2100  
Denmark

Email: anders\_brandt@sigmadesigns.com

Jakob Buron  
Sigma Designs  
Emdrupvej 26A, 1.  
Copenhagen O 2100  
Denmark

Email: jakob\_buron@sigmadesigns.com

IPv6 Maintenance Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 13, 2012

K. Lynn, Ed.  
Consultant  
J. Martocci  
Johnson Controls  
C. Neilson  
Delta Controls  
S. Donaldson  
Honeywell  
March 12, 2012

Transmission of IPv6 over MS/TP Networks  
draft-ietf-6man-6lobac-01

Abstract

MS/TP (Master-Slave/Token-Passing) is a contention-free access method for the RS-485 physical layer that is used extensively in building automation networks. This document describes the frame format for transmission of IPv6 packets and the method of forming link-local and statelessly autoconfigured IPv6 addresses on MS/TP networks.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 13, 2012.

Copyright Notice

Copyright (c) 2012 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 . . . . .	3
2. MS/TP Mode for IPv6 . . . . .	6
3. Addressing Modes . . . . .	6
4. Maximum Transmission Unit (MTU) . . . . .	7
5. LoBAC Adaptation Layer . . . . .	7
6. Stateless Address Autoconfiguration . . . . .	9
7. IPv6 Link Local Address . . . . .	10
8. Unicast Address Mapping . . . . .	10
9. Multicast Address Mapping . . . . .	11
10. Header Compression . . . . .	11
11. IANA Considerations . . . . .	11
12. Security Considerations . . . . .	12
13. Acknowledgments . . . . .	12
14. References . . . . .	12
14.1. Normative References . . . . .	12
14.2. Informative References . . . . .	13
Appendix A. Extended Data CRC [CRC32K] . . . . .	14
Appendix B. Consistent Overhead Byte Stuffing [COBS] . . . . .	16
Authors' Addresses . . . . .	20

## 1. Introduction

MS/TP (Master-Slave/Token-Passing) is a contention-free access method for the RS-485 [TIA-485-A] physical layer that is used extensively in building automation networks. This document describes the frame format for transmission of IPv6 [RFC2460] packets and the method of forming link-local and statelessly autoconfigured IPv6 addresses on MS/TP networks. The general approach is to adapt elements of the 6LoWPAN [RFC4944] specification to constrained wired networks.

An MS/TP device is typically based on a low-cost microcontroller with limited processing power and memory. Together with low data rates and a small address space, these constraints are similar to those faced in 6LoWPAN networks and suggest some elements of that solution might be applied. MS/TP differs significantly from 6LoWPAN in at least three respects: a) MS/TP devices typically have a continuous source of power, b) all MS/TP devices on a segment can communicate directly so there are no hidden node or mesh routing issues, and c) proposed changes to MS/TP will support payloads of up to 1501 octets, eliminating the need for link-layer fragmentation and reassembly.

The following sections provide a brief overview of MS/TP, then describe how to form IPv6 addresses and encapsulate IPv6 packets in MS/TP frames. This document also specifies a header compression mechanism, based on [RFC6282], that is recommended in order to make IPv6 practical on low speed MS/TP networks.

### 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 [RFC2119].

### 1.2. Abbreviations Used

ASHRAE: American Society of Heating, Refrigerating, and Air-Conditioning Engineers (<http://www.ashrae.org>)

BACnet: An ISO/ANSI/ASHRAE Standard Data Communication Protocol for Building Automation and Control Networks

CRC: Cyclic Redundancy Check

MAC: Medium Access Control

MSDU: MAC Service Data Unit (MAC client data)

UART: Universal Asynchronous Transmitter/Receiver

### 1.3. MS/TP Overview

This section provides a brief overview of MS/TP, which is specified in Clause 9 of ANSI/ASHRAE 135-2010 [BACnet] and included herein by reference. [BACnet] also covers physical layer deployment options.

MS/TP is designed to enable multidrop networks over shielded twisted pair wiring. It can support segments up to 1200 meters in length or data rates up to 115,200 baud (at this highest data rate the segment length is limited to 1000 meters). An MS/TP link requires only a UART, a 5ms resolution timer, and a [TIA-485-A] transceiver with a driver that can be disabled. These features combine to make MS/TP a cost-effective field bus for the most numerous and least expensive devices in a building automation network.

The differential signaling used by [TIA-485-A] requires a contention-free MAC. MS/TP uses a token to control access to a multidrop bus. A master node may initiate the transmission of a data frame when it holds the token. After sending at most a configured maximum number of data frames, a master node passes the token to the next master node (as determined by node address). Slave nodes transmit only when polled and are not considered part of this specification.

MS/TP frames have the following format\*:

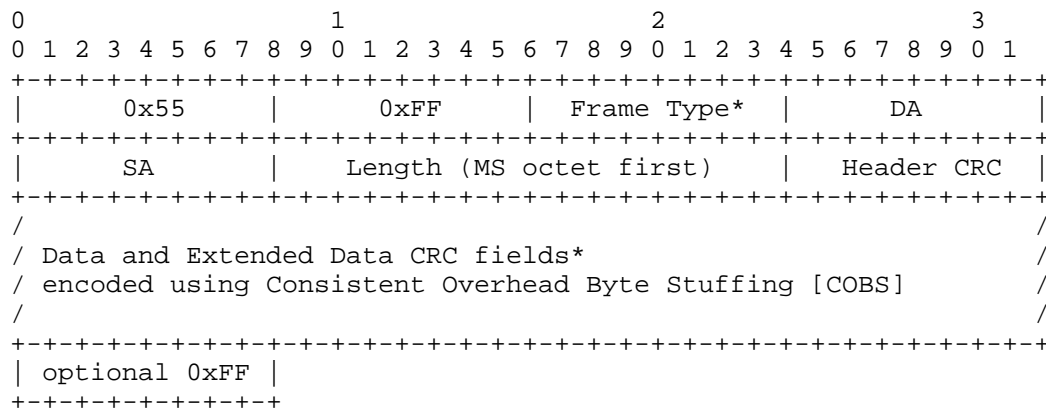


Figure 1: MS/TP Extended Frame Format

\*Note: A BACnet proposal [Addendum\_an], now in public review, assigns a new Frame Type for IPv6 Encapsulation, extends the maximum length of the Data field to 1501 octets, and specifies a 32-bit Extended Data CRC [CRC32K] for these frames. The Data and Extended Data CRC fields are [COBS] encoded and present only if Length is non-zero.

The MS/TP frame fields have the following descriptions\*\*:

Preamble	two octet preamble: 0x55, 0xFF
Frame Type	one octet
Destination Address	one octet address
Source Address	one octet address
Length	two octets, most significant octet first
Header CRC	one octet
Data	0 - 1501 octets** (present only if Length is non-zero)
Extended Data CRC	four octets**, least significant octet first (present only if Length is non-zero)
(pad)	(optional) at most one octet of trailer: 0xFF

The Frame Type is used to distinguish between different types of MAC frames. Currently defined types (in decimal) are:

- 00 Token
- 01 Poll For Master
- 02 Reply To Poll For Master
- ...
- 10 IPv6 over MS/TP Encapsulation\*\*

Frame Types 11 through 127 are reserved for assignment by ASHRAE. All master nodes MUST understand Token, Poll For Master, and Reply to Poll For Master frames. See Section 2 for additional details.

The Destination and Source Addresses are each one octet in length. See Section 3 for additional details.

A non-zero Length field specifies the length of the [COBS] encoded Data and Extended Data CRC fields in octets, minus two. (Note: This trick is required for co-existence with legacy MS/TP devices.) See Section 4 and Appendices for additional details.

The Header CRC field covers the Frame Type, Destination Address, Source Address, and Length fields. The Header CRC generation and check procedures are specified in [BACnet].

\*\*The Data and Extended Data CRC fields are conditional on the Frame Type and the Length and will always be present in frames specified by this document. These fields are concatenated and then encoded before transmission using Consistent Overhead Byte Stuffing [COBS] to remove preamble sequences from the fields. The Extended Data CRC and COBS encoding procedures are specified in the BACnet [Addendum\_an] change proposal and briefly summarized in Appendices A and B below.



#### 1.4. Goals and Non-goals

The primary goal of this specification is to enable IPv6 directly to wired end devices in building automation and control networks, while leveraging existing standards to the greatest extent possible. A secondary goal is to co-exist with legacy MS/TP implementations. Only the minimum changes necessary to support IPv6 over MS/TP are proposed in BACnet [Addendum\_an] (see note in Section 1.3).

Non-goals include making changes to the MS/TP frame header format, control frames, Master Node state machine, or addressing modes. Also, while the techniques described here may be applicable to other data links, no attempt is made to define a general design pattern.

#### 2. MS/TP Mode for IPv6

The BACnet [Addendum\_an] change proposal allocates a new MS/TP Frame Type from the ASHRAE reserved range to indicate IPv6 Encapsulation. The new Frame Type for IPv6 Encapsulation is 10 (0x0A).

All MS/TP master nodes (including those that support IPv6) must understand Token, Poll For Master, and Reply to Poll For Master control frames and support the Master Node state machine as specified in [BACnet]. MS/TP master nodes that support IPv6 must also support the Receive Frame state machine as specified in [BACnet] as extended by [Addendum\_an].

#### 3. Addressing Modes

MS/TP node (link-layer) addresses are one octet in length. The method of assigning node addresses is outside the scope of this document. However, each MS/TP node on the link MUST have a unique address or a misconfiguration condition exists.

[BACnet] specifies that addresses 0 through 127 are valid for master nodes. The method specified in Section 6 for creating the Interface Identifier (IID) ensures that an IID of all zeros can never result.

A Destination Address of 255 (0xFF) denotes a link-level broadcast (all nodes). A Source Address of 255 MUST NOT be used. MS/TP does not support multicast, therefore all IPv6 multicast packets MUST be sent as link-level broadcasts and filtered at the IPv6 layer.

This document assumes that each MS/TP link maps onto a unique IPv6 subnet prefix. Hosts learn IPv6 prefixes via router advertisements according to [RFC4861].

#### 4. Maximum Transmission Unit (MTU)

The BACnet [Addendum\_an] change proposal specifies that the MSDU be increased to 1501 octets and covered by a 32-bit CRC. This is sufficient to convey an MTU of at least 1280 octets as required by IPv6 without the need for link-layer fragmentation and reassembly.

However, the relatively low data rates of MS/TP still make a compelling case for header compression. An adaptation layer to indicate compressed or uncompressed IPv6 headers is specified below in Section 5 and the compression scheme is specified in Section 10.

#### 5. LoBAC Adaptation Layer

The encapsulation formats defined in this section (subsequently referred to as the "LoBAC" encapsulation) comprise the payload (MSDU) of an MS/TP frame. The LoBAC payload (i.e., an IPv6 packet) follows an encapsulation header stack. LoBAC is a subset of the LOWPAN encapsulation defined in [RFC4944], therefore the use of "LOWPAN" in literals below is intentional. The primary differences between LoBAC and LOWPAN are: a) exclusion of the Fragmentation, Mesh, and Broadcast headers, and b) use of LOWPAN\_IPHC [RFC6282] in place of LOWPAN\_HC1 header compression (which is deprecated by [RFC6282]).

All LoBAC encapsulated datagrams transmitted over MS/TP are prefixed by an encapsulation header stack. Each header in the stack consists of a header type followed by zero or more header fields. Whereas in an IPv6 header the stack would contain, in the following order, addressing, hop-by-hop options, routing, fragmentation, destination options, and finally payload [RFC2460]; in a LoBAC encapsulation the analogous sequence is (optional) header compression and payload. The header stacks that are valid in a LoBAC network are shown below.

A LoBAC encapsulated IPv6 datagram:

```
+-----+-----+-----+
| IPv6 Dispatch | IPv6 Header | Payload |
+-----+-----+-----+
```

A LoBAC encapsulated LOWPAN\_IPHC compressed IPv6 datagram:

```
+-----+-----+-----+
| IPHC Dispatch | IPHC Header | Payload |
+-----+-----+-----+
```

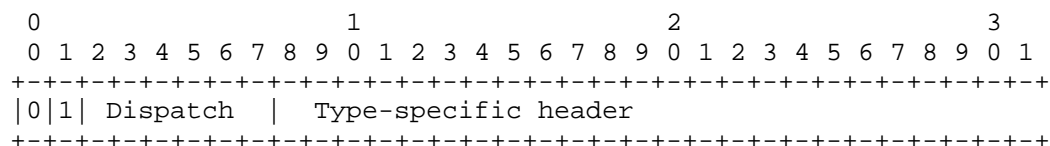
All protocol datagrams (i.e., IPv6 or compressed IPv6 headers) SHALL be preceded by one of the valid LoBAC encapsulation headers. This

permits uniform software treatment of datagrams without regard to their mode of transmission.

The definition of LoBAC headers consists of the dispatch value, the definition of the header fields that follow, and their ordering constraints relative to all other headers. Although the header stack structure provides a mechanism to address future demands on the LoBAC (LoWPAN) adaptation layer, it is not intended to provide general purpose extensibility. This format document specifies a small set of header types using the header stack for clarity, compactness, and orthogonality.

### 5.1. Dispatch Value and Header

The LoBAC Dispatch value begins with a "0" bit followed by a "1" bit. The Dispatch value and header are shown here:



Dispatch	6-bit selector. Identifies the type of header immediately following the Dispatch value.
----------	---

Type-specific header	A header determined by the Dispatch value.
----------------------	--

Figure 2: Dispatch Value and Header

The Dispatch value may be treated as an unstructured namespace. Only a few symbols are required to represent current LoBAC functionality. Although some additional savings could be achieved by encoding additional functionality into the dispatch value, these measures would tend to constrain the ability to address future alternatives.

Pattern		Header Type	
00	xxxxxxx	NALP	- Not a LoWPAN (LoBAC) frame
01	000000	ESC	- Additional Dispatch octet follows
01	000001	IPv6	- Uncompressed IPv6 Addresses
...		reserved	- Defined or reserved by [RFC4944]
01	1xxxxxx	LOWPAN_IPHC	- LOWPAN_IPHC compressed IPv6 [RFC6282]
1x	xxxxxxx	reserved	- Defined or reserved by [RFC4944]

Figure 3: Dispatch Value Bit Patterns

NALP: Specifies that the following bits are not a part of the LoBAC encapsulation, and any LoBAC node that encounters a Dispatch value of 00xxxxxx shall discard the packet. Non-LoBAC protocols that wish to coexist with LoBAC nodes should include an octet matching this pattern immediately following the MS/TP header.

ESC: Specifies that the following header is a single 8-bit field for the Dispatch value. It allows support for Dispatch values larger than 127 (see [RFC6282] section 5).

IPv6: Specifies that the following header is an uncompressed IPv6 header [RFC2460].

LOWPAN\_IPHC: A value of 011xxxxx specifies a LOWPAN\_IPHC compression header (see Section 10.)

Reserved: A LoBAC node that encounters a Dispatch value in the range 01000010 through 01011111 or 1xxxxxxx SHALL discard the packet.

## 6. Stateless Address Autoconfiguration

This section defines how to obtain an IPv6 Interface Identifier. The general procedure is described in Appendix A of [RFC4291], "Creating Modified EUI-64 Format Interface Identifiers".

The Interface Identifier may be based on an [EUI-64] identifier assigned to the device (but this is not typical for MS/TP). In this case, the Interface Identifier is formed from the EUI-64 by inverting the "u" (universal/local) bit according to [RFC4291]. This will result in a globally unique Interface Identifier.

If the device does not have an EUI-64, then the Interface Identifier MUST be formed by concatenating its 8-bit MS/TP node address to the seven octets 0x00, 0x00, 0x00, 0xFF, 0xFE, 0x00, 0x00. For example, an MS/TP node address of hexadecimal value 0x4F results in the following Interface Identifier:

0	1 1	3 3	4 4	6
0	5 6	1 2	7 8	3
+-----+-----+-----+-----+-----+				
0000000000000000 0000000011111111 1111111000000000 0000000001001111				
+-----+-----+-----+-----+-----+				

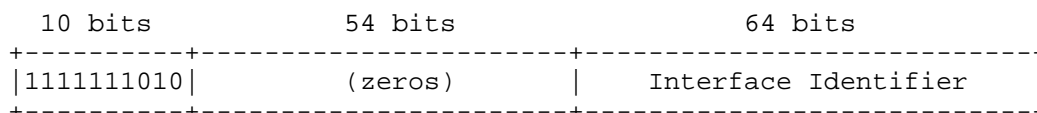
Note that this results in the universal/local bit set to "0" to indicate local scope.

An IPv6 address prefix used for stateless autoconfiguration [RFC4862]

of an MS/TP interface MUST have a length of 64 bits.

## 7. IPv6 Link Local Address

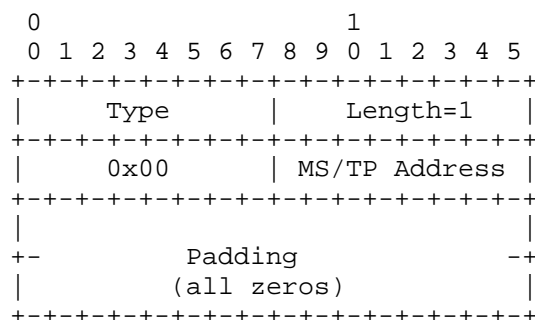
The IPv6 link-local address [RFC4291] for an MS/TP interface is formed by appending the Interface Identifier, as defined above, to the prefix FE80::/64.



## 8. Unicast Address Mapping

The address resolution procedure for mapping IPv6 non-multicast addresses into MS/TP link-layer addresses follows the general description in Section 7.2 of [RFC4861], unless otherwise specified.

The Source/Target Link-layer Address option has the following form when the addresses are 8-bit MS/TP node (link-layer) addresses.



Option fields:

Type:

1: for Source Link-layer address.

2: for Target Link-layer address.

Length: This is the length of this option (including the type and length fields) in units of 8 octets. The value of this field is 1 for 8-bit MS/TP node addresses.

MS/TP Address: The 8-bit address in canonical bit order [RFC2469].  
This is the unicast address the interface currently responds to.

## 9. Multicast Address Mapping

All IPv6 multicast packets MUST be sent to MS/TP Destination Address 255 (broadcast) and filtered at the IPv6 layer. When represented as a 16-bit address in a compressed header (see Section 10), it MUST be formed by padding on the left with a zero:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|           0x00           | 0xFF       |
+---+---+---+---+---+---+---+---+

```

## 10. Header Compression

LoBAC uses LOWPAN\_IPHC IPv6 compression, which is specified in [RFC6282] and included herein by reference. This section will simply identify substitutions that should be made when interpreting the text of [RFC6282].

In general the following substitutions should be made:

- \* Replace "6LoWPAN" with "MS/TP network"
- \* Replace "IEEE 802.15.4 address" with "MS/TP address"

When a 16-bit address is called for (i.e., an IEEE 802.15.4 "short address") it MUST be formed by padding the MS/TP address to the left with a zero:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|           0x00           | MS/TP address |
+---+---+---+---+---+---+---+---+

```

## 11. IANA Considerations

This document uses values previously reserved by [RFC4944] and [RFC6282] and makes no further requests of IANA.

Note to RFC Editor: this section may be removed upon publication.

## 12. Security Considerations

The method of deriving Interface Identifiers from MAC addresses is intended to preserve global uniqueness when possible. However, there is no protection from duplication through accident or forgery.

## 13. Acknowledgments

We are grateful to the authors of [RFC4944] and members of the IETF 6LoWPAN working group; this document borrows extensively from their work.

## 14. References

### 14.1. Normative References

- [BACnet] American Society of Heating, Refrigerating, and Air-Conditioning Engineers, "BACnet, A Data Communication Protocol for Building Automation and Control Networks (ANSI Approved)", ANSI/ASHRAE 135-2010, April 2011.
- [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.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

### 14.2. Informative References

- [Addendum\_an] ASHRAE, "BSR/ASHRAE Addendum an to ANSI/ASHRAE Standard 135-2010, BACnet - A Data Communication Protocol for Building Automation and Control Networks (Advisory Public Review Draft)", September 2011, <<https://osr.ashrae.org/default.aspx>>.
- [COBS] Cheshire, S. and M. Baker, "Consistent Overhead Byte Stuffing", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL.7, NO.2 , April 1999, <<http://www.stuartcheshire.org/papers/COBSforToN.pdf>>.
- [CRC32K] Koopman, P., "32-Bit Cyclic Redundancy Codes for Internet Applications", IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2002) , June 2002, <[http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02\\_koopman.pdf](http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf)>.
- [EUI-64] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", March 1997, <<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>>.
- [RFC2469] Narten, T. and C. Burton, "A Caution On The Canonical Ordering Of Link-Layer Addresses", RFC 2469, December 1998.
- [TIA-485-A] Telecommunications Industry Association, "TIA-485-A, Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems (ANSI/TIA/EIA-485-A-98) (R2003)", March 2003.



## Appendix A. Extended Data CRC [CRC32K]

This Appendix is informative and not part of the standard.

Extending the payload of MS/TP to 1501 octets requires upgrading the Data CRC from 16 bits to 32 bits. Koopman has published several papers on choosing the right CRC polynomial for the application. In [CRC32K], he surveyed the entire 32-bit polynomial space and identified some that exceed the 802.3 polynomial in performance.

The BACnet MS/TP change proposal [Addendum\_an] specifies the CRC32K (Koopman) polynomial. An example C implementation is shown below. The specified use of the function is that 'crcValue' is initialized to all ones before the function is first called and, upon running the function over all octets in the payload, the ones complement of 'crcValue' be sent in LSB-first order. Upon reception, the data field and modified 'crcValue' are passed again through the function. If these fields were properly received, the result of the function will be 'CRC32K\_RESIDUE'.

```
#include <stdint.h>

/* See ASHRAE 135-2010 Addendum an, section G.3.2 */
#define CRC32K_INITIAL_VALUE (0xFFFFFFFF)
#define CRC32K_RESIDUE (0x0843323B)
/* CRC-32K polynomial, 1 + x**1 + ... + x**30 (+ x**32) */
#define CRC32K_POLY (0xEB31D82E)

/*
 * Accumulate 'dataValue' into the CRC in 'crcValue'.
 * Return updated CRC.
 *
 * Note: crcValue must be set to CRC32K_INITIAL_VALUE
 * before initial call.
 */
uint32_t
CalcExtendedDataCRC(uint8_t dataValue, uint32_t crcValue)
{
    uint8_t data, b;
    uint32_t crc;

    data = dataValue;
    crc = crcValue;

    for (b = 0; b < 8; b++) {
        if ((data & 1) ^ (crc & 1)) {
            crc >>= 1;
            crc ^= CRC32K_POLY;
        } else {
            crc >>= 1;
        }
        data >>= 1;
    }
    return crc;
}
```

## Appendix B. Consistent Overhead Byte Stuffing [COBS]

This Appendix is informative and not part of the standard.

The BACnet change proposal [Addendum\_an] corrects a long-standing issue with the MS/TP specification; namely that preamble sequences were not escaped whenever they appeared in the Data or Data CRC fields. In some cases, this could result in dropped frames due to mis-alignment. The solution is encode the Data and Extended Data CRC fields before transmission using Consistent Overhead Byte Stuffing [COBS] and decode these fields upon reception.

COBS is a run-length encoding method that effectively removes '0x00' octets from its input. The worst-case overhead is bounded at approx. one octet in 254, or less than 0.5%, as described in [COBS]. An arbitrary octet value may be removed by XOR'ing the COBS output with the specified value. In the case of MS/TP, the '0x55' preamble octet is specified for removal.

Encoding proceeds logically in three passes. First, the Extended Data CRC is calculated over the data, modified for transmission as described in Appendix A, and appended to the data. The combined fields are then passed through the COBS encoder. The resulting output is then XOR'd with the MS/TP preamble octet '0x55'. The length of the encoded fields, minus two octets for compatibility with existing MS/TP devices, is placed in the MS/TP header Length field before transmission.

An example C implementation that combines these passes is shown below. The decode() function is the inverse of the encode() function.

```
#include <stdint.h>

#define MSTP_PREAMBLE_55 (0x55)

/*
 * Encodes 'length' octets of data at the location pointed to by 'input'
 * and writes the output to the location pointed to by 'output'.
 * Returns the number of octets written to 'output'.
 */
size_t
frame_encode (uint8_t *output, const uint8_t *input, size_t length)
{
    size_t code_index = 0;
    size_t read_index = 0;
    size_t write_index = 1;
    uint8_t code = 1;
    uint8_t data;
    int i;

    uint32_t crc32K = CRC32K_INITIAL_VALUE;

    while (read_index < length) {
        data = input[read_index++];
        crc32K = CalcExtendedDataCRC(data, crc32K);
        /*
         * In the common case, simply copy input to output and
         * increment the number of octets copied.
         */
        if (data != 0) {
            output[write_index++] = (data ^ MSTP_PREAMBLE_55);
            code++;
            if (code != 0xFF)
                continue;
        }
        /*
         * In the special case of encountering a zero in the input or
         * having copied the maximum number (254) of non-zero octets,
         * store the count and re-initialize encoder variables.
         */
        output[code_index] = (code ^ MSTP_PREAMBLE_55);
        code_index = write_index++;
        code = 1;
    }
    /*
     * Run the one's complement of the CRC value through the encoder,
     * LSB first.
     */
    crc32K = ~crc32K;
```

```
for (i = 0; i < 4; i++) {
    data = ((uint8_t *) &crc32K)[i];

    if (data != 0) {
        output[write_index++] = (data ^ MSTP_PREAMBLE_55);
        code++;
        if (code != 0xFF)
            continue;
    }
    /*
     * In the special case of encountering a zero in the input or
     * having copied the maximum number (254) of non-zero octets,
     * store the count and re-initialize encoder variables.
     */
    output[code_index] = (code ^ MSTP_PREAMBLE_55);
    code_index = write_index++;
    last_code = code;
    code = 1;
}
/* Append a "phantom zero" to the output stream. */
output[code_index] = (code ^ MSTP_PREAMBLE_55);
/*
 * Return the combined value of the Data and Extended CRC fields.
 * Subtract two before use as the MS/TP frame Length field.
 */
return write_index;
}
```

```
/*
 * Takes COBS encoded Data and Extended Data CRC fields as 'input'.
 * The 'length' contains the actual length of these fields in
 * octets (that is, the header Length field plus two).
 * Decodes the Data and Extended Data CRC fields into 'output'.
 * Returns length of decoded Data in octets.
 * Note: Safe to call with 'output' <= 'input'.
 */
size_t
frame_decode (uint8_t *output, const uint8_t *input, size_t length)
{
    uint16_t read_index = 0;
    uint16_t write_index = 0;
    uint8_t code, data;
    int i;

    crc32 = CRC32K_INITIAL_VALUE;
    while (read_index < length) {
        code = (input[read_index] ^ MSTP_PREAMBLE_55);
        /*
         * Sanity check the encoding to prevent the for() loop below
         * from overrunning the output buffer.
         */
        if ((read_index + code) > length)
            return 0;

        read_index++;

        for (i = 1; i < code; i++) {
            data = (input[read_index++] ^ MSTP_PREAMBLE_55);
            crc32 = CalcExtendedDataCRC(data, crc32);
            output[write_index++] = data;
        }
        if ((code < 0xFF) && (read_index < length)) {
            data = '\0';
            crc32 = CalcExtendedDataCRC(data, crc32);
            output[write_index++] = data;
        }
    }
    if (crc32K == CRC32K_RESIDUE)
        return write_index - sizeof(uint32_t);
    else
        return 0;
}
```

Authors' Addresses

Kerry Lynn (editor)  
Consultant

Phone: +1 978 460 4253  
Email: kerlyn@ieee.org

Jerry Martocci  
Johnson Controls, Inc.  
507 E. Michigan St  
Milwaukee, WI 53202  
USA

Phone: +1 414 524 4010  
Email: jerald.p.martocci@jci.com

Carl Neilson  
Delta Controls, Inc.  
17850 56th Ave  
Surrey, BC V3S 1C7  
Canada

Phone: +1 604 575 5913  
Email: cneilson@deltaccontrols.com

Stuart Donaldson  
Honeywell Automation & Control Solutions  
6670 185th Ave NE  
Redmond, WA 98052  
USA

Email: stuart.donaldson@honeywell.com





6LoWPAN  
Internet-Draft  
Intended status: Informational  
Expires: January 16, 2014

P. Mariager, Ed.  
J. Petersen  
RTX A/S  
Z. Shelby  
Sensinode  
July 15, 2013

Transmission of IPv6 Packets over DECT Ultra Low Energy  
draft-mariager-6lowpan-v6over-dect-ule-03

Abstract

DECT Ultra Low Energy is a low power air interface technology that is defined by the DECT Forum and specified by ETSI.

The DECT air interface technology has been used world-wide in communication devices for more than 15 years, primarily carrying voice for cordless telephony but has also been deployed for data centric services.

The DECT Ultra Low Energy is a recent addition to the DECT interface primarily intended for low-bandwidth, low-power applications such as sensor devices, smart meters, home automation etc. As the DECT Ultra Low Energy interface inherits many of the capabilities from DECT, it benefits from long range, interference free operation, world wide reserved frequency band, low silicon prices and maturity. There is an added value in the ability to communicate with IPv6 over DECT ULE such as for Internet of Things applications.

This document describes how IPv6 is transported over DECT ULE using 6LoWPAN techniques.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 16, 2014.

#### Copyright Notice

Copyright (c) 2013 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 . . . . .	2
1.1. Requirements Notation . . . . .	3
1.2. Terms Used . . . . .	3
2. DECT Ultra Low Energy . . . . .	4
2.1. The DECT ULE Protocol Stack . . . . .	4
2.2. Link layer roles and topology . . . . .	6
2.3. Addressing Model . . . . .	7
2.4. MTU Considerations . . . . .	7
2.5. Additional Considerations . . . . .	8
3. Specification of IPv6 over DECT ULE . . . . .	8
3.1. Protocol stack . . . . .	8
3.2. Link model . . . . .	8
3.3. Internet connectivity scenarios . . . . .	11
4. IANA Considerations . . . . .	12
5. Security Considerations . . . . .	13
6. ETSI Considerations . . . . .	13
7. Acknowledgements . . . . .	13
8. Normative References . . . . .	13
Authors' Addresses . . . . .	14

#### 1. Introduction

DECT Ultra Low Energy (DECT ULE or just ULE) is an air interface technology building on the key fundamentals of traditional DECT / CAT-iq but with specific changes to significantly reduce the power consumption on the expense of data throughput. DECT ULE devices with requirements to power consumption will operate on special power optimized silicon, but can connect to a DECT Gateway supporting traditional DECT / CAT-iq for cordless telephony and data as well as

the ULE extensions. DECT terminology operates with two major role definitions: The Portable Part (PP) is the power constrained device, while the Fixed Part (FP) is the Gateway or base station. This FP may be connected to the Internet. An example of a use case for DECT ULE is a home security sensor transmitting small amounts of data (few bytes) at periodic intervals through the FP, but is able to wake up upon an external event (burglar) and communicate with the FP. Another example incorporating both DECT ULE as well as traditional CAT-iq telephony is an elderly pendant (broche) which can transmit periodic status messages to a care provider using very little battery, but in the event of urgency, the elderly person can establish a voice connection through the pendant to an alarm service. It is expected that DECT ULE will be integrated into many residential gateways, as many of these already implements DECT CAT-iq for cordless telephony. DECT ULE can be added as a software option for the FP. It is desirable to consider IPv6 for DECT ULE devices due to the large address space and well-known infrastructure. This document describes how IPv6 is used on DECT ULE links to optimize power while maintaining the many benefits of IPv6 transmission. [RFC4944] specifies the transmission of IPv6 over IEEE 802.15.4. DECT ULE has in many ways similar characteristics of IEEE 802.15.4, but also differences. Many of the mechanisms defined in [RFC4944] can be applied to the transmission of IPv6 on DECT ULE links.

This document specifies how to map IPv6 over DECT ULE inspired by RFC4944

### 1.1. Requirements Notation

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

### 1.2. Terms Used

PP: DECT Portable Part, typically the sensor node

FP: DECT Fixed Part, the gateway

LLME: Lower Layer Management Entity

NWK: Network

## 2. DECT Ultra Low Energy

DECT ULE is a low power air interface technology that is designed to support both circuit switched for service, such as voice communication, and for packet mode data services at modest data rate. This draft is only addressing the packet mode data service of DECT ULE.

### 2.1. The DECT ULE Protocol Stack

The DECT ULE protocol stack consists of the PHY layer operating at frequencies in the 1880 - 1920 MHz frequency band depending on the region and uses a symbol rate of 1.152 Mbps. Radio bearers are allocated by use of FDMA/TDMA/TDD technics.

In its generic network topology, DECT is defined as a cellular network technology. However, the most common configuration is a star network with a single FP defining the network with a number of PP attached. The MAC layer supports both traditional DECT as this is used for services like discovery, pairing, security features etc. All these features have been reused from DECT.

The DECT ULE device can then switch to the ULE mode of operation, utilizing the new ULE MAC layer features. The DECT ULE Data Link Control (DLC) provides multiplexing as well as segmentation and re-assembly for larger packets from layers above. The DECT ULE layer also implements per-message authentication and encryption. The DLC layer ensures packet integrity and preserves packet order, but delivery is based on best effort.

The current DECT ULE MAC layer standard supports low bandwidth data broadcast. However the usage of this broadcast service has not yet been standardized for higher layers and no security has been developed been developed yet. This document is not considering usage of this DECT ULE MAC broadcast service in current version.

In general, communication sessions can be initiated from both FP and PP side. Depending of power down modes employed in the PP, latency may occur when initiating sessions from FP side. MAC layer communication can either take place using connection oriented packet transfer with low overhead for short sessions or take place using connection oriented bearers including media reservation. The MAC layer autonomously selects the radio spectrum positions that are available within the band and can rearrange these to avoid interference. The MAC layer has built-in retransmission procedures in order to improve transmission reliability.

The DECT ULE device will typically incorporate an Application Programmers Interface (API) as well as common elements known as Generic Access Profile (GAP) for enrolling into the network. The DECT ULE stack establishes a permanent virtual circuit (PVC) for the application layers and provides support for a range of different application protocols. The used application protocol is negotiated between the PP and FP when the PVC communication service is established. This draft proposes to define 6LoWPAN as one of the possible protocols to negotiate.

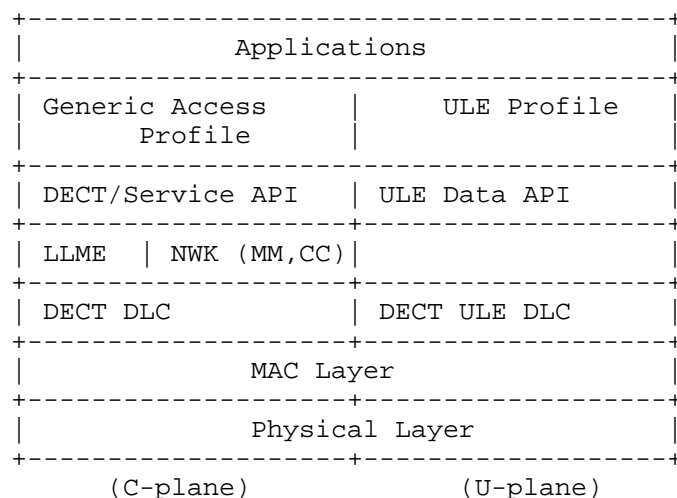


Figure 1: DECT ULE Protocol Stack

The DECT ULE stack can be divided into control (C-plane) and user-data (U-plane) parts shown to the left and to the right in figure 1, respectively.

It is expected that the ULE 6LoWPAN adaptation layer can run directly on this U-plane DLC layer. Figure 2 illustrates IPv6 over DECT ULE stack.

Constrained Application Protocol (CoAP) is an application protocol specifically designed for resource constrained environments. CoAP could be run on top of IPv6 supporting requests from the server and requests of cached replies from a CoAP/HTTP proxy in the DECT Fixed Part or in an external network infrastructure.

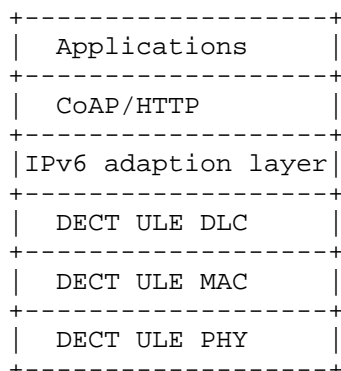


Figure 2: IPv6 over DECT ULE Stack

## 2.2. Link layer roles and topology

A FP is assumed to be less constrained than a PP. Hence, in the primary scenario FP and PP will act as 6LoWPAN Border Router (6LBR) and a 6LoWPAN Node (6LN), respectively. This document does only address this primary scenario.

In DECT ULE, communication only takes place between a FP and a PP. A FP is able to handle multiple simultaneous connections with a number of PP. Hence, in a DECT ULE network using IPv6, a radio hop is equivalent to an IPv6 link and vice versa.

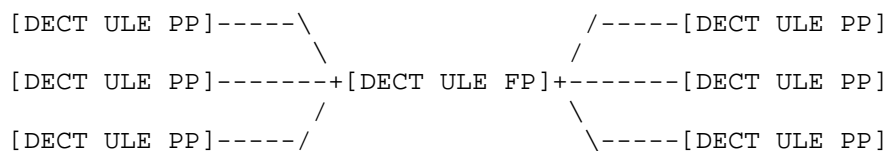


Figure 3: DECT ULE star topology

DECT ULE repeaters are not considered in this proposal.

### 2.3. Addressing Model

Each DECT PP is assigned an <IPEI> (International Portable Equipment Identity) during manufacturing. This identity has the size of 40 bits and is globally unique for the PP and can be used to constitute the MAC address.

When bound to a FP, a PP is assigned a 20 bit TPUI (Temporary Portable User Identity) which is unique within the FP. This TPUI is used for addressing (layer 2) in messages between FP and PP.

Each DECT FP is assigned a <RFPI> (Radio Fixed Part Identity) during manufacturing. This identity has the size of 40 bits and is globally unique for a FP and can be used to constitute the MAC address.

Alternatively each DECT PP and DECT FP can be assigned a unique (IEEE) MAC-48 address additionally to the DECT identities to be used by the 6LoWPAN. When such approach, the FP and PP have to implement a mapping between used MAC-48 addresses and DECT identities.

### 2.4. MTU Considerations

Generally the DECT ULE FP and PP may be generating data that fits into one MAC Layer packet (38 bytes) for periodically transferred information, depending on application. IP data packets may be much larger and hence MTU size should be the size of the IP data packet. The DECT ULE DLC procedures supports segmentation and reassembly of any MTU size below 65536 bytes, but most implementations do only support smaller values.

If an implementation cannot support the sufficient MTU size (due to implementation cost) then SAR needs to be supported at upper layers. The SAR feature of [RFC4944] section 5 could be considered.

It is expected that the LOWPAN\_IPHC packet will fulfill all the requirements for header compression without spending unnecessary overhead for mesh addressing.

It is important to realize that the support of larger packets will be on the expense of battery life, as a large packet will be fragmented into several or many MAC layer packets, each consuming power to transmit / receive.

## 2.5. Additional Considerations

The DECT ULE standard allows PP to be registered (bind) to multiple FP and roaming between these FP. This draft does not consider the scenarios of PP roaming between multiple FP. The use of repeater functionality is also not considered in this draft.

## 3. Specification of IPv6 over DECT ULE

DECT ULE technology sets strict requirements for low power consumption and thus limits the allowed protocol overhead. 6LoWPAN standard [RFC4944] provides useful functionality for reducing overhead which can be applied to DECT ULE. This functionality comprises of link-local IPv6 addresses and stateless IPv6 address autoconfiguration, Neighbor Discovery and header compression.

A significant difference between IEEE 802.15.4 and DECT ULE is that the former supports both star and mesh topology (and requires a routing protocol), whereas DECT ULE in its primary configuration does not support the formation of multihop networks at the link layer. In consequence, the mesh header defined in [RFC4944] for mesh under routing MUST NOT be used in DECT ULE networks. In addition, a DECT ULE PP node MUST NOT play the role of a 6LoWPAN Router (6LR).

### 3.1. Protocol stack

DECT ULE standardization of protocol identifier in negotiation of higher layer application protocol 6LoWPAN: xx. This identifier is reserved for 6LoWPAN and has to be standardized by ETSI.

### 3.2. Link model

The general model is that IPv6 is layer 3 and DECT ULE MAC+DLC is layer 2. The DECT ULE implements FAR functionality and RFC4944 MUST NOT be used. Since IPv6 requires MTU size of at least 1280 octets, the DECT ULE connection (PVC) must be configured with equivalent MTU size.

This specification also assumes the IPv6 header compression format specified in [RFC6282]. It is also assumed that the IPv6 payload length can be inferred from the ULE DLC packet length and the IID value inferred from the link-layer address.



Due to DECT ULE star topology, each branch of the star is considered to be an individual link and thus the PP cannot directly hear each other and also cannot talk to each other with link-local addresses. After the FP and PP have connected at the DECT ULE level, the link can be considered up and IPv6 address configuration and transmission can begin. The FP ensures address collisions do not occur.

### 3.2.1. IPv6 Address Configuration

A DECT ULE 6LN performs stateless address autoconfiguration as per RFC 4862. A 64-bit Interface identifier (IID) for a DECT ULE interface MAY be formed by utilizing a MAC-48 device address as defined in RFC 2464 "IPv6 over Ethernet" specification. Alternatively, the DECT device addresses IPEI, RFPI or TPUI, MAY be used instead to derive the IID. In the case of randomly generated IID or use of IID derived from DECT devices addresses, the "Universal/Local" bit MUST be set to 0. Only if a global unique MAC-48 is used the "Universal/Local" bit can be set to 1.

As defined in RFC 4291, the IPv6 link-local address for a DECT ULE node is formed by appending the IID, to the prefix FE80::/64.

The means for a 6LBR to obtain an IPv6 prefix for numbering the DECT ULE network is out of scope of this document, but can be, for example, accomplished via DHCPv6 Prefix Delegation or by using Unique Local IPv6 Unicast Addresses (ULA). Due to the link model of the DECT ULE the 6LBR MUST set the "on-link" flag (L) to zero in the Prefix Information Option. This will cause 6LNs to always send packets to the 6LBR, including the case when the destination is another 6LN using the same prefix.

### 3.2.2. Neighbor discovery

'Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)' [RFC6775] describes the neighbor discovery approach as adapted for use in several 6LoWPAN topologies, including the mesh topology. As DECT ULE is considered not to support mesh networks, hence only those aspects that apply to a star topology are considered.

The following aspects of the Neighbor Discovery optimizations [RFC6775] are applicable to DECT ULE 6LNs:

1. A DECT ULE 6LN MUST register its address with the 6LBR by sending a Neighbor Solicitation (NS) message with the ARO option and process the Neighbor Advertisement (NA) accordingly. The NS with the ARO option SHOULD be sent irrespective of whether the IID is derived from a unique MAC-48 bit device address, DECT ULE device addresses or the

IID is a random value that is generated as per the privacy extensions for stateless address autoconfiguration [RFC4941]. Although RFC 4941 [RFC4941] permits the use of deprecated addresses for old connections, in this specification we mandate that one interface MUST NOT use more than one IID at any one time.

2. For sending Router Solicitations and processing Router Advertisements the DECT ULE 6LNs MUST, respectively, follow Sections 5.3 and 5.4 of the [RFC6775].

### 3.2.3. Unicast and Multicast address mapping

The DECT MAC layer broadcast service is considered inadequate for IP multicast.

Hence traffic is always unicast between two DECT ULE nodes. Even in the case where a FP is attached to multiple PPs, the FP cannot do a multicast to all the connected PPs. If the FP needs to send a multicast packet to all its PPs, it has to replicate the packet and unicast it on each link. However, this may not be energy-efficient and particular care must be taken if the FP is battery-powered. In the opposite direction, a PPs can only transmit data to a single destination (i.e. the FP). Hence, when a PP needs to transmit an IPv6 multicast packet, the PP will unicast the corresponding DECT ULE packet to the FP. As described in the linkmodel section FP will not forward link-local multicast messages to other PPs connected to the FP.

### 3.2.4. Header Compression

Header compression as defined in RFC 6282, which specifies the compression format for IPv6 datagrams on top of IEEE 802.15.4, is REQUIRED in this document as the basis for IPv6 header compression on top of DECT ULE. All headers MUST be compressed according to RFC 6282 encoding formats. The DECT ULE's star topology structure can be exploited in order to provide a mechanism for IID compression. The following text describes the principles of IPv6 address compression on top of DECT ULE.

In a link-local communication, both the IPv6 source and destination addresses MUST be elided, since the node knows that the packet is destined for it even if the packet does not have destination IPv6 address. On the other hand, a node SHALL learn the IID of the other endpoint of each DECT ULE connection it participates in. By exploiting this information, a node that receives a data channel PDU containing an IPv6 packet can infer the corresponding IPv6 source address. A node MUST maintain a Neighbor Cache, in which the entries include both the IID of the neighbor and the Device Address that

identifies the neighbor. For the type of communication considered in this paragraph, the following settings MUST be used in the IPv6 compressed header: CID=0, SAC=0, SAM=11, DAC=0, DAM=11.

When a 6LN transmits an IPv6 packet to a remote destination using global Unicast IPv6 addresses, if a context is defined for the prefix of the 6LN's global IPv6 address, the 6LN MUST indicate this context in the corresponding source fields of the compressed IPv6 header as per Section 3.1 of RFC 6282, and MUST elide the IPv6 source address. For this, the 6LN MUST use the following settings in the IPv6 compressed header: CID=1, SAC=1, SAM=11. In this case, the 6LBR can infer the elided IPv6 source address since 1) the 6LBR has previously assigned the prefix to the 6LNs; and 2) the 6LBR maintains a Neighbor Cache that relates the Device Address and the IID of the corresponding PP. If a context is defined for the IPv6 destination address, the 6LN MUST also indicate this context in the corresponding destination fields of the compressed IPv6 header, and MUST elide the prefix of the destination IPv6 address. For this, the 6LN MUST set the DAM field of the compressed IPv6 header as DAM=01 (if the context covers a 64-bit prefix) or as DAM=11 (if the context covers a full, 128-bit address). CID and DAC MUST be set to CID=1 and DAC=1. Note that when a context is defined for the IPv6 destination address, the 6LBR can infer the elided destination prefix by using the context.

When a 6LBR receives an IPv6 packet sent by a remote node outside the DECT ULE network, and the destination of the packet is a 6LN, if a context is defined for the prefix of the 6LN's global IPv6 address, the 6LBR MUST indicate this context in the corresponding destination fields of the compressed IPv6 header, and MUST elide the IPv6 destination address of the packet before forwarding it to the 6LN. For this, the 6LBR MUST set the DAM field of the IPv6 compressed header as DAM=11. CID and DAC MUST be set to CID=1 and DAC=1. If a context is defined for the prefix of the IPv6 source address, the 6LBR MUST indicate this context in the source fields of the compressed IPv6 header, and MUST elide that prefix as well. For this, the 6LBR MUST set the SAM field of the IPv6 compressed header as SAM=01 (if the context covers a 64-bit prefix) or SAM=11 (if the context covers a full, 128-bit address). CID and SAC MUST be set to CID=1 and SAC=1.

### 3.3. Internet connectivity scenarios

In a typical scenario, the DECT ULE network is connected to the Internet as shown in the Figure 4.

A degenerate scenario can be imagined where a PP is acting as 6LBR and providing Internet connectivity for the FP. How the FP could then further provide Internet connectivity to other PP, possibly connected to the FP, is out of the scope of this document.

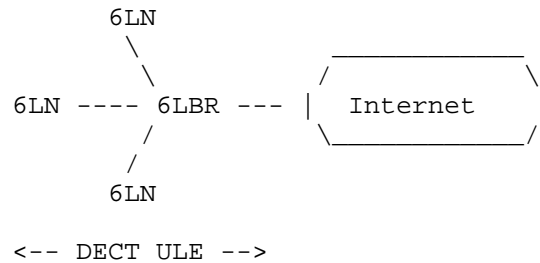


Figure 4: DECT ULE network connected to the Internet

In some scenarios, the DECT ULE network may transiently or permanently be an isolated network as shown in the Figure 5.

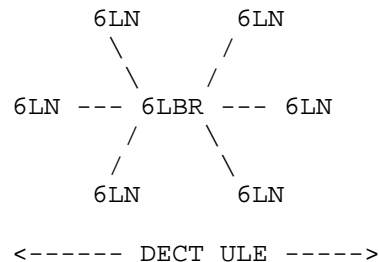


Figure 5: Isolated DECT ULE network

In the isolated network scenario communications between 6LN and 6LBR can use IPv6 link-local methodology, but for communications between different PP, the FP has to act as 6LBR, number the network with ULA prefix [RFC4193], and route packets between PP.

#### 4. IANA Considerations

There are no IANA considerations related to this document.

## 5. Security Considerations

The secure transmission of speech over DECT will be based on the DSAA2 and DSC2 work being developed by the DF Security group / ETSI TC DECT and the ETSI SAGE Security expert group.

DECT ULE communication are secured by encryption and per-message authentication through CCM mode (Counter with CBC-MAC) similar to RFC3610, which has been defined in the ETSI TC-DECT ULE group. DECT ULE DLC layer implements this per-message authentication and encryption to provide link-layer security mechanisms as defined by ETSI TC-DECT.

The underlying algorithm for providing authentication and encryption is based on AES128. Individual authentication key (UAK) for each ULE PP are generated during the binding procedure. Session encryption keys are renewed regularly. DECT ULE PPs do not use any shared encryption key.

The DECT ULE pairing procedure generates a master security key and during location registration procedure or when the permanent virtual circuit are established, the session security keys are generated. The generated security keys are individual for each FP-PP binding, hence all PP in a system have different security keys.

## 6. ETSI Considerations

ETSI is standardizing a list of known application layer protocols that can use the DECT ULE permanent virtual circuit packet data service. Each protocol is identified by a unique known identifier. The IPv6/6LoWPAN as described in this document is considered as an application layer protocol on top of DECT ULE. In order to provide interoperability between 6LoWPAN / DECT ULE devices a common protocol identifier for 6LoWPAN has to be standardized by ETSI.

It is proposed to used ETSI DECT ULE protocol identifier 0x06 = 6LoWPAN.

## 7. Acknowledgements

## 8. Normative References

[ETSI-EN300.175-part1-7]

, .

[ETSI-TS102.939-1]

, .

[I-D.ietf-6lowpan-hc]

, .

[I-D.ietf-6lowpan-nd]

, .

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

[RFC4291] , .

[RFC4944] , .

#### Authors' Addresses

Peter B. Mariager (editor)  
RTX A/S  
Stroemmen 6  
DK-9400 Noerresundby  
Denmark

Email: pm@rtx.dk

Jens Toftgaard Petersen  
RTX A/S  
Stroemmen 6  
DK-9400 Noerresundby  
Denmark

Email: jtp@rtx.dk

Zach Shelby  
Sensinode  
Hallituskatu 13-17D  
FI-90100 Oulu  
Finland

Email: zach.shelby@sensinode.com

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: May 11, 2014

J. Schoenwaelder  
A. Sehgal  
Jacobs University  
T. Tsou  
Huawei Technologies (USA)  
C. Zhou  
Huawei Technologies  
November 7, 2013

Definition of Managed Objects for IPv6 over Low-Power Wireless Personal  
Area Networks (6LoWPANs)  
draft-schoenw-6lo-lowpan-mib-01

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it defines objects for managing IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 11, 2014.

Copyright Notice

Copyright (c) 2013 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 . . . . .	3
2. The Internet-Standard Management Framework . . . . .	3
3. Conventions . . . . .	3
4. Overview . . . . .	3
5. Relationship to Other MIB Modules . . . . .	6
6. Definitions . . . . .	6
7. Security Considerations . . . . .	14
8. IANA Considerations . . . . .	15
9. Acknowledgements . . . . .	15
10. References . . . . .	15
10.1. Normative References . . . . .	15
10.2. Informative References . . . . .	16
Appendix A. JSON Representation . . . . .	17



## 1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols. In particular it defines objects for managing IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) [RFC4944].

## 2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIV2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

## 3. Conventions

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].

## 4. Overview

The LOWPAN-MIB module is primarily a collection of counters that reflect how 6LoWPAN datagrams are processed by the 6LoWPAN layer. The object identifier registration tree has the following structure:

```

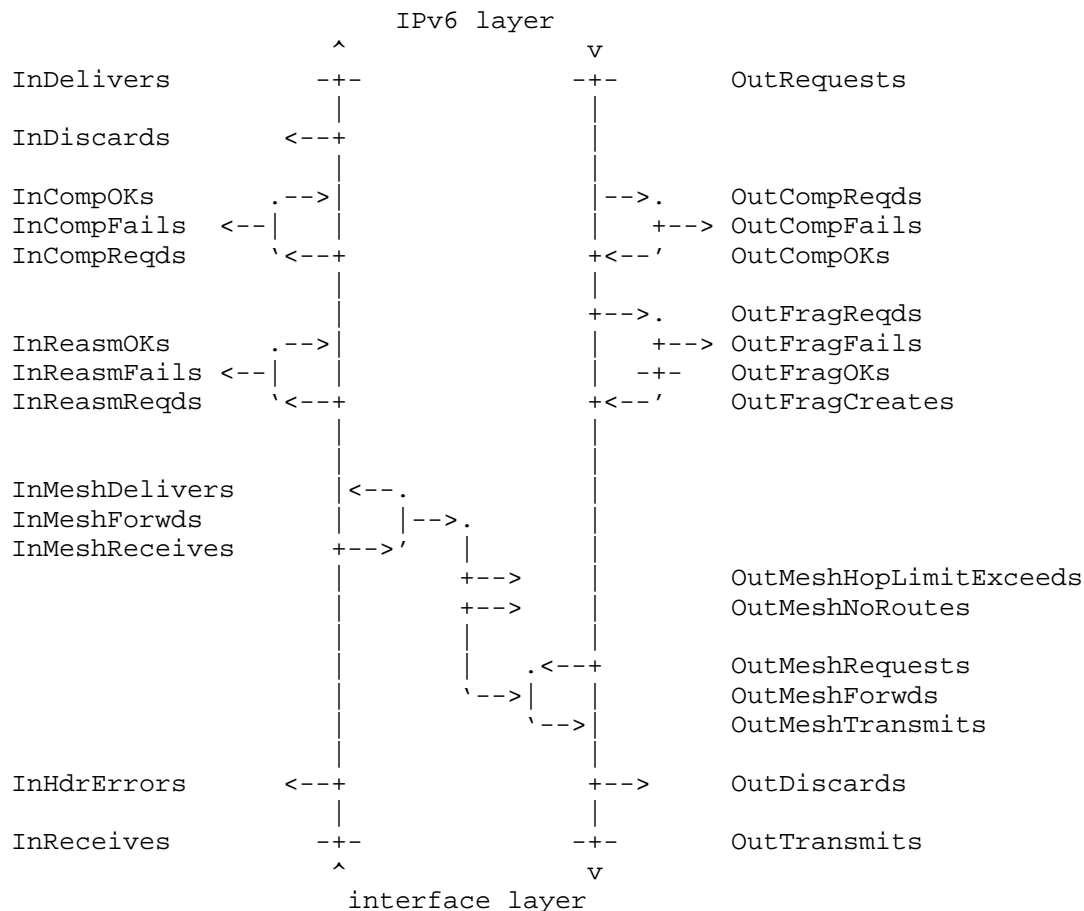
--lowpanMIB(1.3.6.1.2.1.XXXX)
+--lowpanNotifications(0)
+--lowpanObjects(1)
|   +-- r-n Unsigned32 lowpanReasmTimeout(1)
|   +-- r-n Counter32  lowpanInReceives(2)
|   +-- r-n Counter32  lowpanInHdrErrors(3)
|   +-- r-n Counter32  lowpanInMeshReceives(4)
|   +-- r-n Counter32  lowpanInMeshForwds(5)
|   +-- r-n Counter32  lowpanInMeshDelivers(6)
|   +-- r-n Counter32  lowpanInReasmReqds(7)
|   +-- r-n Counter32  lowpanInReasmFails(8)
|   +-- r-n Counter32  lowpanInReasmOKs(9)
|   +-- r-n Counter32  lowpanInCompReqds(10)
|   +-- r-n Counter32  lowpanInCompFails(11)
|   +-- r-n Counter32  lowpanInCompOKs(12)
|   +-- r-n Counter32  lowpanInDiscards(13)
|   +-- r-n Counter32  lowpanInDelivers(14)
|   +-- r-n Counter32  lowpanOutRequests(15)
|   +-- r-n Counter32  lowpanOutCompReqds(16)
|   +-- r-n Counter32  lowpanOutCompFails(17)
|   +-- r-n Counter32  lowpanOutCompOKs(18)
|   +-- r-n Counter32  lowpanOutFragReqds(19)
|   +-- r-n Counter32  lowpanOutFragFails(20)
|   +-- r-n Counter32  lowpanOutFragOKs(21)
|   +-- r-n Counter32  lowpanOutFragCreates(22)
|   +-- r-n Counter32  lowpanOutMeshHopLimitExceeds(23)
|   +-- r-n Counter32  lowpanOutMeshNoRoutes(24)
|   +-- r-n Counter32  lowpanOutMeshRequests(25)
|   +-- r-n Counter32  lowpanOutMeshForwds(26)
|   +-- r-n Counter32  lowpanOutMeshTransmits(27)
|   +-- r-n Counter32  lowpanOutDiscards(28)
|   +-- r-n Counter32  lowpanOutTransmits(29)
+--lowpanConformance(2)
+--lowpanGroups(1)
|   +--lowpanCoreGroup(1)
|   +--lowpanMeshGroup(2)
+--lowpanCompliances(2)
+--lowpanCompliance(1)

```

The counters defined in the LOWPAN-MIB module provide information about the 6LoWPAN datagrams received and transmitted and how they are processed in the 6LoWPAN layer. For link-layers that use the 6LoWPAN dispatch byte as defined in [RFC4944] (e.g., IEEE 802.15.4), a 6LoWPAN datagram is a datagram with a dispatch byte matching the bit patterns 01xxxxxx, 10xxxxxx, or 11xxxxxx. Datagrams with a dispatch byte matching the bit pattern 00xxxxxx (NALP - not a LoWPAN frame) are not considered to be 6LoWPAN datagram by this specification. Other radio technologies may use different mechanisms to identify

6LoWPAN datagrams (e.g., the BLUETOOTH Low Energy Logical Link Control and Adaptation Protocol uses Channel Identifiers [I-D.ietf-6lowpan-btle]).

The following diagram illustrates the conceptual relationships between the counters.



The fragmentation related counters have been modeled after the fragmentation related counters of the IP-MIB [RFC4293]. The discard counters have been placed at the end of the input and output chains but they can be bumped any time if a datagram is discarded for a reason not covered by the other counters.

The compression related counters provide insights into compression requests and in particular also compression related failures. Note that the diagram is conceptual in the sense that compression happens

after reassembly for incoming 6LoWPAN datagrams and compression happens before fragmentation for outgoing 6LoWPAN datagrams. Implementations may choose to implement things slightly differently. For example, implementations may decompress FRAG1 fragments as soon as they are received, not waiting for reassembly to complete.

The mesh header processing related counters do not have an explicit discard counter. Implementations that do not support mesh forwarding MUST count the number of received 6LoWPAN datagrams with a MESH header (lowpanInMeshReceives) but they MUST NOT increment the lowpanInMeshReceives and lowpanInMeshDelivers counters if these 6LoWPAN datagrams are dropped.

## 5. Relationship to Other MIB Modules

The MIB module IMPORTS definitions from SNMPv2-SMI [RFC2578] and SNMPv2-CONF [RFC2580].

## 6. Definitions

LOWPAN-MIB DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE, Unsigned32, Counter32, mib-2  
FROM SNMPv2-SMI -- RFC 2578  
OBJECT-GROUP, MODULE-COMPLIANCE  
FROM SNMPv2-CONF; -- RFC 2580

lowpanMIB MODULE-IDENTITY

LAST-UPDATED "201311070000Z"

ORGANIZATION

"Jacobs University Bremen"

CONTACT-INFO

"Juergen Schoenwaelder

Jacobs University Bremen

Email: j.schoenwaelder@jacobs-university.de

Anuj Sehgal

Jacobs University Bremen

Email: s.anuj@jacobs-university.de

Tina Tsou

Huawei Technologies

Email: tina.tsou.zouting@huawei.com

Cathy Zhou

Huawei Technologies

```
Email: cathyzhou@huawei.com"
DESCRIPTION
"The MIB module for monitoring nodes implementing the IPv6
over Low-Power Wireless Personal Area Networks (6LoWPAN)
protocol.

Copyright (c) 2013 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD
License set forth in Section 4.c of the IETF Trust's
Legal Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info)."
```

```
REVISION "201311070000Z"
DESCRIPTION
"Initial version, published as RFC XXXX."
-- RFC Ed.: replace XXXX with actual RFC number and remove this note

::= { mib-2 XXXX }

-- object definitions

lowpanNotifications      OBJECT IDENTIFIER ::= { lowpanMIB 0 }
lowpanObjects            OBJECT IDENTIFIER ::= { lowpanMIB 1 }
lowpanConformance       OBJECT IDENTIFIER ::= { lowpanMIB 2 }

lowpanReasmTimeout OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum number of seconds that received fragments are
        held while they are awaiting reassembly at this entity."
    ::= { lowpanObjects 1 }

lowpanInReceives OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of 6LoWPAN datagrams received, including
        those received in error."
    ::= { lowpanObjects 2 }
```

## lowpanInHdrErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of received 6LoWPAN datagrams discarded due to errors in their headers, including unknown dispatch values."

::= { lowpanObjects 3 }

## lowpanInMeshReceives OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of received 6LoWPAN datagrams with a MESH header."

::= { lowpanObjects 4 }

## lowpanInMeshForwds OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of received 6LoWPAN datagrams requiring MESH forwarding."

::= { lowpanObjects 5 }

## lowpanInMeshDelivers OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of received 6LoWPAN datagrams with a MESH header delivered to the local system."

::= { lowpanObjects 6 }

## lowpanInReasmReqds OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of received 6LoWPAN fragments that needed to be reassembled. This includes both FRAG1 and FRAGN 6LoWPAN datagrams."

::= { lowpanObjects 7 }

## lowpanInReasmFails OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

```
STATUS      current
DESCRIPTION
    "The number of failures detected by the re-assembly algorithm
    (e.g., timeouts). Note that this is not necessarily a count of
    discarded 6LoWPAN fragments since implementations can lose
    track of the number of fragments by combining them as
    received."
 ::= { lowpanObjects 8 }

lowpanInReasmOKs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of IPv6 packets successfully reassembled."
    ::= { lowpanObjects 9 }

lowpanInCompReqds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams requiring header
        decompression."
    ::= { lowpanObjects 10 }

lowpanInCompFails OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams where header decompression
        failed (e.g., because the necessary context information was
        not available)."
    ::= { lowpanObjects 11 }

lowpanInCompOKs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams where header decompression
        was successful."
    ::= { lowpanObjects 12 }

lowpanInDiscards OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
```

```
STATUS      current
DESCRIPTION
    "The number of received 6LoWPAN datagrams for which no
    problems were encountered to prevent their continued
    processing, but were discarded (e.g., for lack of buffer
    space). Note that this counter does not include any
    datagrams discarded due to a reassembly failure or a
    compression failure."
 ::= { lowpanObjects 13 }

lowpanInDelivers OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The total number of IPv6 packets successfully delivered
    to the IPv6 layer."
 ::= { lowpanObjects 14 }

lowpanOutRequests OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The total number of IPv6 packets supplied by the IPv6 layer."
 ::= { lowpanObjects 15 }

lowpanOutCompReqds OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The total number of IPv6 packets for which header compression
    was attempted."
 ::= { lowpanObjects 16 }

lowpanOutCompFails OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The total number of IPv6 packets for which header compression
    failed."
 ::= { lowpanObjects 17 }

lowpanOutCompOKs OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
```



```
STATUS      current
DESCRIPTION
    "The total number of IPv6 packets for which header compression
    was successful."
 ::= { lowpanObjects 18 }

lowpanOutFragReqds OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of IPv6 packets that required fragmentation
    in order to be transmitted."
 ::= { lowpanObjects 19 }

lowpanOutFragFails OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of IPv6 packets that have been discarded because
    fragmentation failed."
 ::= { lowpanObjects 20 }

lowpanOutFragOKs OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of IPv6 packets that have been successfully
    fragmented."
 ::= { lowpanObjects 21 }

lowpanOutFragCreates OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of 6LoWPAN fragments that have been
    generated as a result of fragmentation. This includes
    both FRAG1 and FRAGN 6LoWPAN datagrams."
 ::= { lowpanObjects 22 }

lowpanOutMeshHopLimitExceeds OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
```

"The number of 6LoWPAN datagrams with a MESH header that were dropped because the hop limit has been exceeded."  
 ::= { lowpanObjects 23 }

lowpanOutMeshNoRoutes OBJECT-TYPE

SYNTAX Counter32  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"The number of 6LoWPAN datagrams with a MESH header that were dropped because there was no forwarding information available."  
 ::= { lowpanObjects 24 }

lowpanOutMeshRequests OBJECT-TYPE

SYNTAX Counter32  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"The number of 6LoWPAN datagrams requiring MESH header encapsulation."  
 ::= { lowpanObjects 25 }

lowpanOutMeshForwds OBJECT-TYPE

SYNTAX Counter32  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"The number of 6LoWPAN datagrams with a MESH header for which suitable forwarding information was available."  
 ::= { lowpanObjects 26 }

lowpanOutMeshTransmits OBJECT-TYPE

SYNTAX Counter32  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"The number of 6LoWPAN datagrams with a MESH header created."  
 ::= { lowpanObjects 27 }

lowpanOutDiscards OBJECT-TYPE

SYNTAX Counter32  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"The number of IPv6 packets for which no problem was encountered to prevent their transmission to their

```
        destination, but were discarded (e.g., for lack of
        buffer space)."
```

```
 ::= { lowpanObjects 28 }
```

```
lowpanOutTransmits OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of 6LoWPAN datagram that this entity
        supplied to the lower layers for transmission."
    ::= { lowpanObjects 29 }
```

```
-- conformance definitions
```

```
lowpanGroups          OBJECT IDENTIFIER ::= { lowpanConformance 1 }
lowpanCompliances     OBJECT IDENTIFIER ::= { lowpanConformance 2 }
```

```
lowpanCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "Compliance statement for systems that implement 6LoWPAN."
    MODULE      -- this module
    MANDATORY-GROUPS {
        lowpanCoreGroup
    }
    GROUP        lowpanMeshGroup
    DESCRIPTION
        "This group is mandatory for implementations that process
        or forward 6LoWPAN datagrams with mesh headers."
    ::= { lowpanCompliances 1 }
```

```
lowpanCoreGroup OBJECT-GROUP
    OBJECTS {
        lowpanReasmTimeout,
        lowpanInReceives,
        lowpanInHdrErrors,
        lowpanInMeshReceives,
        lowpanInReasmReqds,
        lowpanInReasmFails,
        lowpanInReasmOKs,
        lowpanInCompReqds,
        lowpanInCompFails,
        lowpanInCompOKs,
        lowpanInDiscards,
        lowpanInDelivers,
        lowpanOutRequests,
        lowpanOutCompReqds,
```

```
        lowpanOutCompFails,
        lowpanOutCompOKs,
        lowpanOutFragReqds,
        lowpanOutFragFails,
        lowpanOutFragOKs,
        lowpanOutFragCreates,
        lowpanOutDiscards,
        lowpanOutTransmits
    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing information and
        statistics about the processing of 6LoWPAN datagrams,
        excluding counters covering the processing of datagrams
        with a mesh headers."
    ::= { lowpanGroups 1 }

lowpanMeshGroup OBJECT-GROUP
    OBJECTS {
        lowpanInMeshForwds,
        lowpanInMeshDelivers,
        lowpanOutMeshHopLimitExceeds,
        lowpanOutMeshNoRoutes,
        lowpanOutMeshRequests,
        lowpanOutMeshForwds,
        lowpanOutMeshTransmits
    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing information and
        statistics about the processing of 6LoWPAN datagrams
        with a 6LoWPAN mesh header."
    ::= { lowpanGroups 2 }

END
```

## 7. Security Considerations

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

The read-only counters provide insights into the amount of 6LoWPAN traffic a node is receiving or transmitting. This might provide

information whether a device is regularly exchanging information with other devices or whether a device is mostly not participating in any communication (e.g., the device might be "easier" to take away unnoticed). The reassembly counters could be used to direct denial of service attacks on the reassembly mechanism.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

## 8. IANA Considerations

IANA is requested to assign a value for "XXXX" under the 'mib-2' subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "XXXX" (here and in the MIB module) with the assigned value and to remove this note.

## 9. Acknowledgements

This specification borrows heavily from the IP-MIB defined in [RFC4293].

Juergen Schoenwaelder and Anuj Sehgal were partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

## 10. References

### 10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels",

BCP 14, RFC 2119, March 1997.

- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [I-D.ietf-6lowpan-btle] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets over BLUETOOTH Low Energy", draft-ietf-6lowpan-btle-12 (work in progress), February 2013.

## 10.2. Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.
- [RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.
- [RFC6643] Schoenwaelder, J., "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules", RFC 6643, July 2012.
- [I-D.lhotka-netmod-yang-json] Lhotka, L., "Modeling JSON Text with

YANG",  
draft-lhotka-netmod-yang-json-02 (work  
in progress), September 2013.

#### Appendix A. JSON Representation

Using the translation algorithm defined in [RFC6643], the SMIV2 module can be translated to YANG. Using the JSON representation of data modeled in YANG defined in [I-D.lhotka-netmod-yang-json], the objects defined in the MIB module can be represented in JSON as shown below. The compact representation without any white space uses 468 octets. (Of course, this number depends on the number of octets needed for the counter values.)

```
{
  "LOWPAN-MIB:LOWPAN-MIB": {
    "lowpanReasmTimeout": 20,
    "lowpanInReceives": 42,
    "lowpanInHdrErrors": 0,
    "lowpanInMeshReceives": 8,
    "lowpanInMeshForwds": 0,
    "lowpanInMeshDelivers": 0,
    "lowpanInReasmReqds": 22,
    "lowpanInReasmFails": 2,
    "lowpanInReasmOKs": 20,
    "lowpanInCompReqds": 16,
    "lowpanInCompFails": 2,
    "lowpanInCompOKs": 14,
    "lowpanInDiscards": 1,
    "lowpanInDelivers": 12,
    "lowpanOutRequests": 12,
    "lowpanOutCompReqds": 0,
    "lowpanOutCompFails": 0,
    "lowpanOutCompOKs": 0,
    "lowpanOutFragReqds": 5,
    "lowpanOutFragFails": 0,
    "lowpanOutFragOKs": 5,
    "lowpanOutFragCreates": 8,
    "lowpanOutMeshHopLimitExceeds": 0,
    "lowpanOutMeshNoRoutes": 0,
    "lowpanOutMeshRequests": 0,
    "lowpanOutMeshForwds": 0,
    "lowpanOutMeshTransmits": 0,
    "lowpanOutDiscards": 0,
    "lowpanOutTransmits": 15
  }
}
```

Authors' Addresses

Juergen Schoenwaelder  
Jacobs University  
Campus Ring 1  
Bremen 28759  
Germany

EMail: j.schoenwaelder@jacobs-university.de

Anuj Sehgal  
Jacobs University  
Campus Ring 1  
Bremen 28759  
Germany

EMail: s.anuj@jacobs-university.de

Tina Tsou  
Huawei Technologies (USA)  
2330 Central Expressway  
Santa Clara CA 95050  
USA

EMail: tina.tsou.zouting@huawei.com

Cathy Zhou  
Huawei Technologies  
Bantian, Longgang District  
Shenzhen 518129  
P.R. China

EMail: cathyzhou@huawei.com





6lo  
Internet-Draft  
Updates: 4944 (if approved)  
Intended status: Standards Track  
Expires: July 20, 2018

P. Thubert, Ed.  
Cisco Systems  
J. Hui  
Nest Labs  
January 16, 2018

LLN Fragment Forwarding and Recovery  
draft-thubert-6lo-forwarding-fragments-08

Abstract

Considering that an LLN frame can have a MAC payload below 100 bytes, an IPv6 packet might be fragmented into more than 10 fragments at the 6LoWPAN layer. In a 6LoWPAN mesh-under network, the fragments can be forwarded individually across the mesh, whereas a route-over mesh network, a fragmented 6LoWPAN packet must be reassembled at every hop, which causes latency and congestion. This draft introduces a simple protocol to forward individual fragments across a route-over mesh network, and, regardless of the type of mesh, recover the loss of individual fragments across the mesh and protect the network against bloat with a minimal flow control.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 20, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Updating RFC 4944 . . . . .	3
3. Terminology and Referenced Work . . . . .	4
4. New Dispatch types and headers . . . . .	5
4.1. Recoverable Fragment Dispatch type and Header . . . . .	5
4.2. RFRAG Acknowledgment Dispatch type and Header . . . . .	7
5. Fragments Recovery . . . . .	9
6. Forwarding Fragments . . . . .	10
6.1. Upon the first fragment . . . . .	11
6.2. Upon the next fragments . . . . .	12
6.3. Upon the RFRAG Acknowledgments . . . . .	12
7. Security Considerations . . . . .	13
8. IANA Considerations . . . . .	13
9. Acknowledgments . . . . .	13
10. References . . . . .	13
10.1. Normative References . . . . .	13
10.2. Informative References . . . . .	14
Appendix A. Rationale . . . . .	15
Appendix B. Requirements . . . . .	17
Appendix C. Considerations On Flow Control . . . . .	18
Authors' Addresses . . . . .	19

## 1. Introduction

In most Low Power and Lossy Network (LLN) applications, the bulk of the traffic consists of small chunks of data (in the order few bytes to a few tens of bytes) at a time. Given that an IEEE Std. 802.15.4 [IEEE.802.15.4] frame can carry 74 bytes or more in all cases, fragmentation is usually not required. However, and though this happens only occasionally, a number of mission critical applications do require the capability to transfer larger chunks of data, for instance to support a firmware upgrades of the LLN nodes or an extraction of logs from LLN nodes. In the former case, the large chunk of data is transferred to the LLN node, whereas in the latter, the large chunk flows away from the LLN node. In both cases, the size can be on the order of 10Kbytes or more and an end-to-end reliable transport is required.

"Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] defines the original 6LoWPAN datagram fragmentation mechanism for LLNs. One critical issue with this original design is that routing an IPv6 [RFC8200] packet across a route-over mesh requires to reassemble the full packet at each hop, which may cause latency along a path and an overall buffer bloat in the network. Those undesirable effects can be alleviated by a hop-by-hop fragment forwarding technique such as the one proposed in this specification, and arguably this could be achieved without the need to define a new protocol. However, adding that capability alone to the local implementation of the original 6LoWPAN fragmentation would not address the bulk of the issues raised against it, and may create new issues like uncontrolled state in the network.

Another issue against RFC 4944 [RFC4944] is that it does not define a mechanism to first discover the loss of a fragment along a multi-hop path (e.g. having exhausted the link-layer retries at some hop on the way), and then to recover that loss. With RFC 4944, the forwarding of a whole datagram fails when one fragment is not delivered properly to the destination 6LoWPAN endpoint. End-to-end transport or application-level mechanisms may require a full retransmission of the datagram, wasting resources in an already constrained network.

In that situation, the source 6LoWPAN endpoint will not be aware that a loss occurred and will continue sending all fragments for a datagram that is already doomed. The original support is missing signaling to abort a multi-fragment transmission at any time and from either end, and, if the capability to forward fragments is implemented, clean up the related state in the network. It is also lacking flow control capabilities to avoid participating to a congestion that may in turn cause the loss of a fragment and trigger the retransmission of the full datagram.

This specification proposes a method to forward fragments across a multi-hop route-over mesh, and to recover individual fragments between LLN endpoints. The method is designed to limit congestion loss in the network and addresses the requirements that are detailed in Appendix B.

## 2. Updating RFC 4944

This specification updates the fragmentation mechanism that is specified in "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] for use in route-over LLNs by providing a model where fragments can be forwarded end-to-end across a 6LoWPAN LLN, and where fragments that are lost on the way can be recovered individually. New dispatch types are defined in Section 4.

### 3. Terminology and Referenced Work

Past experience with fragmentation has shown that miss-associated or lost fragments can lead to poor network behavior and, occasionally, trouble at application layer. The reader is encouraged to read "IPv4 Reassembly Errors at High Data Rates" [RFC4963] and follow the references for more information.

That experience led to the definition of "Path MTU discovery" [RFC8201] (PMTUD) protocol that limits fragmentation over the Internet.

Specifically in the case of UDP, valuable additional information can be found in "UDP Usage Guidelines for Application Designers" [RFC8085].

Readers are expected to be familiar with all the terms and concepts that are discussed in "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

"The Benefits of Using Explicit Congestion Notification (ECN)" [RFC8087] provides useful information on the potential benefits and pitfalls of using ECN.

Quoting the "Multiprotocol Label Switching (MPLS) Architecture" [RFC3031]: with MPLS, "packets are "labeled" before they are forwarded. At subsequent hops, there is no further analysis of the packet's network layer header. Rather, the label is used as an index into a table which specifies the next hop, and a new label". The MPLS technique is leveraged in the present specification to forward fragments that actually do not have a network layer header, since the fragmentation occurs below IP.

This specification uses the following terms:

#### 6LoWPAN endpoints

The LLN nodes in charge of generating or expanding a 6LoWPAN header from/to a full IPv6 packet. The 6LoWPAN endpoints are the points where fragmentation and reassembly take place.

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

#### 4. New Dispatch types and headers

This specification enables the 6LoWPAN fragmentation sublayer to provide an MTU up to 2048 bytes to the upper layer, which can be the 6LoWPAN Header Compression sublayer that is defined in the "Compression Format for IPv6 Datagrams" [RFC6282] specification. In order to achieve this, this specification enables the fragmentation and the reliable transmission of fragments over a multihop 6LoWPAN mesh network.

This specification provides a technique that is derived from MPLS in order to forward individual fragments across a 6LoWPAN route-over mesh. The datagram\_tag is used as a label; it is locally unique to the node that is the source MAC address of the fragment, so together the MAC address and the label can identify the fragment globally. A node may build the datagram\_tag in its own locally-significant way, as long as the selected tag stays unique to the particular datagram for the lifetime of that datagram. It results that the label does not need to be globally unique but also that it must be swapped at each hop as the source MAC address changes.

This specification extends RFC 4944 [RFC4944] with 4 new Dispatch types, for Recoverable Fragment (RFRAG) headers with or without Acknowledgment Request (RFRAG vs. RFRAG-ARQ), and for the RFRAG Acknowledgment back, with or without ECN Echo (RFRAG-ACK vs. RFRAG-ECHO).

(to be confirmed by IANA) The new 6LoWPAN Dispatch types use the Value Bit Pattern of 11 1010xx from page 0 [RFC8025], as follows:

Pattern	Header Type
11 101000	RFRAG - Recoverable Fragment
11 101001	RFRAG-ARQ - RFRAG with Ack Request
11 101010	RFRAG-ACK - RFRAG Acknowledgment
11 101011	RFRAG-ECHO - RFRAG Ack with ECN Echo

Figure 1: Additional Dispatch Value Bit Patterns

##### 4.1. Recoverable Fragment Dispatch type and Header

In this specification, the size and offset of the fragments are expressed on the compressed packet form as opposed to the uncompressed - native - packet form.

The first fragment is recognized by a sequence of 0; it carries its fragment\_size and the datagram\_size of the compressed packet, whereas

the other fragments carry their `fragment_size` and `fragment_offset`. The last fragment for a datagram is recognized when its `fragment_offset` and its `fragment_size` add up to the `datagram_size`.

Recoverable Fragments are sequenced and a bitmap is used in the RFRAG Acknowledgment to indicate the received fragments by setting the individual bits that correspond to their sequence.

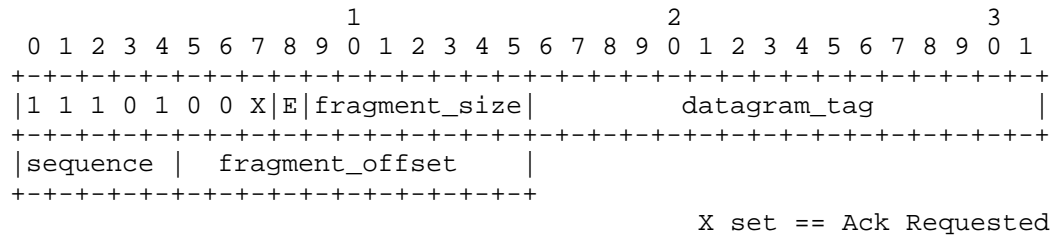


Figure 2: RFRAG Dispatch type and Header

X: 1 bit; Ack Requested: when set, the sender requires an RFRAG Acknowledgment from the receiver.

E: 1 bit; Explicit Congestion Notification; the "E" flag is reset by the source of the fragment and set by intermediate routers to signal that this fragment experienced congestion along its path.

Fragment\_size: 7 bit unsigned integer; the size of this fragment in a unit that depends on the MAC layer technology. For IEEE Std. 802.15.4, the unit is octet, and the maximum fragment size, which is constrained by the maximum frame size of 128 octet minus the overheads of the MAC and Fragment Headers, is not limited by this encoding.

Sequence: 5 bit unsigned integer; the sequence number of the fragment. Fragments are sequence numbered [0..N] where N is in [0..31]. As long as the overheads enable a fragment size of 64 octets or more, this enables to fragment a packet of 2047 octets.

Fragment\_offset: 11 bit unsigned integer;

- \* When set to a non-0 value, the semantics of the `Fragment_offset` depends on the value of the `Sequence`.
- + When the `Sequence` is not 0, this field indicates the offset of the fragment in the compressed form. The fragment should be forwarded based on an existing Label Switched Path (LSP) as described in Section 6.2, or silently dropped if none is found.

- + When the Sequence is 0, denoting the first fragment of a datagram, this field is overloaded to indicate the `total_size` of the compressed packet, to help the receiver allocate an adapted buffer for the reception and reassembly operations. This format limits the maximum MTU on a 6LoWPAN link to 2047 bytes, but 1280 bytes is the recommended value to avoid issues with IPV6 Path MTU Discovery [RFC8201]. The fragment should be routed based on the destination IPv6 address, and an LSP state should be installed as described in Section 6.1.
- \* When set to 0, this field indicates an abort condition and all state regarding the datagram should be cleaned up once the processing of the fragment is complete; the processing of the fragment depends on whether there is an LSP already established for this datagram, and the next hop is still reachable:
  - + if an LSP already exists and is not broken, the fragment is to be forwarded along that LSP as described in Section 6.2, but regardless of the value of the Sequence field;
  - + else, if the Sequence is 0, then the fragment is to be routed as described in Section 6.1 but no state is conserved afterwards.

If the fragment cannot be forwarded or routed, then it is silently dropped.

#### 4.2. RFRAG Acknowledgment Dispatch type and Header

This specification also defines a 4-octet RFRAG Acknowledgment bitmap that is used by the reassembling end point to confirm selectively the reception of individual fragments. A given offset in the bitmap maps one to one with a given sequence number.

The offset of the bit in the bitmap indicates which fragment is acknowledged as follows:



Figure 3: RFRAG Acknowledgment bitmap encoding

Figure 4: Expanding 3 octets encoding

Figure 5: RFRAG Acknowledgment Dispatch type and Header

When set, the sender indicates that at least one of the acknowledged fragments was received with an Explicit Congestion Notification, indicating that the path followed by the fragments is subject to congestion.

An RFRAG Acknowledgment Bitmap, whereby setting the bit at offset *x* indicates that fragment *x* was received, as shown in Figure 3. All 0's is a NULL bitmap that indicates that the fragmentation process is aborted. All 1's is a FULL bitmap that indicates that the fragmentation process is complete, all fragments were received at the reassembly end point.

## 5. Fragments Recovery

The Recoverable Fragment headers RFRAG and RFRAG-ARQ are used to transport a fragment and optionally request an RFRAG Acknowledgment that will confirm the good reception of a one or more fragments. An RFRAG Acknowledgment can optionally carry an ECN indication; it is carried as a standalone header in a message that is sent back to the 6LoWPAN endpoint that was the source of the fragments, as known by its MAC address. The process ensures that at every hop, the source MAC address and the datagram\_tag in the received fragment are enough information to send the RFRAG Acknowledgment back towards the source 6LoWPAN endpoint by reversing the MPLS operation.

The 6LoWPAN endpoint that fragments the packets at 6LoWPAN level (the sender) also controls when the reassembling end point sends the RFRAG Acknowledgments by setting the Ack Requested flag in the RFRAG packets. It may set the Ack Requested flag on any fragment to perform congestion control by limiting the number of outstanding fragments, which are the fragments that have been sent but for which reception or loss was not positively confirmed by the reassembling endpoint. When the sender of the fragment knows that an underlying link-layer mechanism protects the Fragments, it may refrain from using the RFRAG Acknowledgment mechanism, and never set the Ack Requested bit. When it receives a fragment with the ACK Request flag set, the 6LoWPAN endpoint that reassembles the packets at 6LoWPAN level (the receiver) sends back an RFRAG Acknowledgment to confirm reception of all the fragments it has received so far.

The sender transfers a controlled number of fragments and MAY flag the last fragment of a series with an RFRAG Acknowledgment Request. The receiver MUST acknowledge a fragment with the acknowledgment request bit set. If any fragment immediately preceding an acknowledgment request is still missing, the receiver MAY intentionally delay its acknowledgment to allow in-transit fragments to arrive. Delaying the acknowledgment might defeat the round trip delay computation so it should be configurable and not enabled by default.

The receiver MAY issue unsolicited acknowledgments. An unsolicited acknowledgment signals to the sender endpoint that it can resume sending if it had reached its maximum number of outstanding

fragments. Another use is to inform that the reassembling endpoint has cancelled the process of an individual datagram. Note that acknowledgments might consume precious resources so the use of unsolicited acknowledgments should be configurable and not enabled by default.

An observation is that streamlining forwarding of fragments generally reduces the latency over the LLN mesh, providing room for retries within existing upper-layer reliability mechanisms. The sender protects the transmission over the LLN mesh with a retry timer that is computed according to the method detailed in [RFC6298]. It is expected that the upper layer retries obey the recommendations in "UDP Usage Guidelines" [RFC8085], in which case a single round of fragment recovery should fit within the upper layer recovery timers.

Fragments are sent in a round robin fashion: the sender sends all the fragments for a first time before it retries any lost fragment; lost fragments are retried in sequence, oldest first. This mechanism enables the receiver to acknowledge fragments that were delayed in the network before they are actually retried.

When the sender decides that a packet should be dropped and the fragmentation process canceled, it sends a pseudo fragment with the `fragment_offset`, `sequence` and `fragment_size` all set to 0, and no data. Upon reception of this message, the receiver should clean up all resources for the packet associated to the `datagram_tag`. If an acknowledgment is requested, the receiver responds with a NULL bitmap.

The receiver might need to cancel the process of a fragmented packet for internal reasons, for instance if it is out of reassembly buffers, or considers that this packet is already fully reassembled and passed to the upper layer. In that case, the receiver SHOULD indicate so to the sender with a NULL bitmap. Upon an acknowledgment with a NULL bitmap, the sender MUST abort the current fragmented transmission of the datagram.

## 6. Forwarding Fragments

It is assumed that the first Fragment is large enough to carry the IPv6 header and make routing decisions. If that is not so, then this specification MUST NOT be used.

This specification enables intermediate routers to forward fragments with no intermediate reconstruction of the entire packet. The first fragment carries the IP header and it is routed all the way from the fragmenting end point to the reassembling end point. Upon the first fragment, the routers along the path install a label-switched path

(LSP), and the following fragments are label-switched along that path. As a consequence, alternate routes not possible for individual fragments. The datagram\_tag is used to carry the label, that is swapped at each hop. All fragments follow the same path and fragments are delivered in the order at which they are sent.

#### 6.1. Upon the first fragment

In Route-Over mode, the source and destination MAC addressed in a frame change at each hop. The label that is formed and placed in the datagram\_tag is associated to the source MAC and only valid (and unique) for that source MAC. Say the first fragment has:

- o Source IPv6 address = IP\_A (maybe hops away)
- o Destination IPv6 address = IP\_B (maybe hops away)
- o Source MAC = MAC\_previous
- o Datagram\_tag= DT\_previous

The intermediate router that forwards individual fragments performs the following action:

1. a route lookup to get the Next hop IPv6 towards IP\_B, which resolves as IP\_next.
2. a MAC address resolution to get the MAC address associated to IP\_next, which resolves as MAC\_next

Since it is a first fragment of a packet from that source MAC address MAC\_previous for that tag DT\_previous, the router:

1. cleans up any leftover resource associated to the tuple (MAC\_previous, DT\_previous)
2. allocates a new label for that flow, DT\_next, from a Least Recently Used pool or some similar procedure.
3. allocates an abstract label-swap entry indexed by (MAC\_previous, DT\_previous) that contains (MAC\_next, DT\_next)
4. allocates a reflective abstract label-swap structure indexed by (MAC\_next, DT\_next) that contains (MAC\_previous, DT\_previous); this enables the reverse MPLS switching operation that is used to route the RFRAG-ACK.
5. change the source MAC address from MAC\_prev to MAC\_self

6. change the destination MAC address to from MAC\_self to MAC\_next
7. Swaps the datagram\_tag to DT\_next

At this point the router is all set and can forward the fragment to next.

#### 6.2. Upon the next fragments

Upon next fragments (that are not first fragment), the router expects to have already installed a label-swap structure indexed by (MAC\_previous, DT\_previous). The router:

1. looks up the label-swap entry for (MAC\_previous, DT\_previous), which resolves as (MAC\_next, DT\_next)
2. swaps the MAC info to from self to MAC\_next;
3. Swaps the datagram\_tag to DT\_next

if the label-swap entry for (MAC\_previous, DT\_previous) is not found, the router builds an RFRAG-ACK to indicate the error. The resulting message has the following information:

- o MAC info set to from self to MAC\_previous as found in the fragment
- o The datagram\_tag set to DT\_previous
- o Null bitmap to indicate the error

At this point the router is all set and can send the RFRAG-ACK back to the previous router.

#### 6.3. Upon the RFRAG Acknowledgments

Upon an RFRAG Acknowledgment, the router expects to already have label-swap structure indexed by (MAC\_next, DT\_next), which are respectively the source MAC address of the received frame and the received datagram\_tag. DT\_next should have been computed by this router and this router should have assigned it to this particular datagram. The router:

1. looks up the label-swap entry for (MAC\_next, DT\_next), which resolves as (MAC\_previous, DT\_previous)
2. swaps the MAC info to from self to MAC\_previous;
3. Swaps the datagram\_tag to DT\_previous

At this point the router is all set and can forward the RFRAG-ACK to previous.

If the label-swap entry for (MAC\_next, DT\_next) is not found, it MUST silently drop the packet.

If the RFRAG-ACK indicates either an error (NULL bitmap) or that the fragment was entirely received (FULL bitmap), the router schedules the label-swap entries for recycling. If the RFRAG-ACK is lost on the way back, the source may retry the last fragment, which will result as an error RFRAG-ACK from the first router on the way that has already cleaned up.

## 7. Security Considerations

The process of recovering fragments does not appear to create any opening for new threat compared to "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

## 8. IANA Considerations

Need extensions for formats defined in "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

## 9. Acknowledgments

The author wishes to thank Thomas Watteyne and Michael Richardson for in-depth reviews and comments. Also many thanks to Jay Werb, Christos Polyzois, Soumitri Kolavennu, Pat Kinney, Margaret Wasserman, Richard Kelsey, Carsten Bormann and Harry Courtice for their various contributions.

## 10. References

### 10.1. Normative References

- [IEEE.802.15.4]  
IEEE, "IEEE Standard for Low-Rate Wireless Networks",  
IEEE Standard 802.15.4, DOI 10.1109/IEEESTD.2016.7460875,  
<<http://ieeexplore.ieee.org/document/7460875/>>.
- [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>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [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>>.

## 10.2. Informative References

- [I-D.ietf-6tisch-architecture] Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-13 (work in progress), November 2017.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, DOI 10.17487/RFC4919, August 2007, <<https://www.rfc-editor.org/info/rfc4919>>.

- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <<https://www.rfc-editor.org/info/rfc4963>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<https://www.rfc-editor.org/info/rfc7554>>.
- [RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <<https://www.rfc-editor.org/info/rfc7567>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <<https://www.rfc-editor.org/info/rfc8087>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

#### Appendix A. Rationale

There are a number of uses for large packets in Wireless Sensor Networks. Such usages may not be the most typical or represent the largest amount of traffic over the LLN; however, the associated functionality can be critical enough to justify extra care for ensuring effective transport of large packets across the LLN.

The list of those usages includes:



Towards the LLN node:

Firmware update: For example, a new version of the LLN node software is downloaded from a system manager over unicast or multicast services. Such a reflashing operation typically involves updating a large number of similar LLN nodes over a relatively short period of time.

Packages of Commands: A number of commands or a full configuration can be packaged as a single message to ensure consistency and enable atomic execution or complete roll back. Until such commands are fully received and interpreted, the intended operation will not take effect.

From the LLN node:

Waveform captures: A number of consecutive samples are measured at a high rate for a short time and then transferred from a sensor to a gateway or an edge server as a single large report.

Data logs: LLN nodes may generate large logs of sampled data for later extraction. LLN nodes may also generate system logs to assist in diagnosing problems on the node or network.

Large data packets: Rich data types might require more than one fragment.

Uncontrolled firmware download or waveform upload can easily result in a massive increase of the traffic and saturate the network.

When a fragment is lost in transmission, the lack of recovery in the original fragmentation system of RFC 4944 implies that all fragments are resent, further contributing to the congestion that caused the initial loss, and potentially leading to congestion collapse.

This saturation may lead to excessive radio interference, or random early discard (leaky bucket) in relaying nodes. Additional queuing and memory congestion may result while waiting for a low power next hop to emerge from its sleeping state.

Considering that RFC 4944 defines an MTU is 1280 bytes and that in most incarnations (but 802.15.4g) a IEEE Std. 802.15.4 frame can limit the MAC payload to as few as 74 bytes, a packet might be fragmented into at least 18 fragments at the 6LoWPAN shim layer. Taking into account the worst-case header overhead for 6LoWPAN Fragmentation and Mesh Addressing headers will increase the number of required fragments to around 32. This level of fragmentation is much higher than that traditionally experienced over the Internet with

IPv4 fragments. At the same time, the use of radios increases the probability of transmission loss and Mesh-Under techniques compound that risk over multiple hops.

Mechanisms such as TCP or application-layer segmentation could be used to support end-to-end reliable transport. One option to support bulk data transfer over a frame-size-constrained LLN is to set the Maximum Segment Size to fit within the link maximum frame size. Doing so, however, can add significant header overhead to each 802.15.4 frame. In addition, deploying such a mechanism requires that the end-to-end transport is aware of the delivery properties of the underlying LLN, which is a layer violation, and difficult to achieve from the far end of the IPv6 network.

## Appendix B. Requirements

For one-hop communications, a number of Low Power and Lossy Network (LLN) link-layers propose a local acknowledgment mechanism that is enough to detect and recover the loss of fragments. In a multihop environment, an end-to-end fragment recovery mechanism might be a good complement to a hop-by-hop MAC level recovery. This draft introduces a simple protocol to recover individual fragments between 6LoWPAN endpoints that may be multiple hops away. The method addresses the following requirements of a LLN:

### Number of fragments

The recovery mechanism must support highly fragmented packets, with a maximum of 32 fragments per packet.

### Minimum acknowledgment overhead

Because the radio is half duplex, and because of silent time spent in the various medium access mechanisms, an acknowledgment consumes roughly as many resources as data fragment.

The new end-to-end fragment recovery mechanism should be able to acknowledge multiple fragments in a single message and not require an acknowledgment at all if fragments are already protected at a lower layer.

### Controlled latency

The recovery mechanism must succeed or give up within the time boundary imposed by the recovery process of the Upper Layer Protocols.

### Optional congestion control

The aggregation of multiple concurrent flows may lead to the saturation of the radio network and congestion collapse.

The recovery mechanism should provide means for controlling the number of fragments in transit over the LLN.

#### Appendix C. Considerations On Flow Control

Considering that a multi-hop LLN can be a very sensitive environment due to the limited queuing capabilities of a large population of its nodes, this draft recommends a simple and conservative approach to congestion control, based on TCP congestion avoidance.

Congestion on the forward path is assumed in case of packet loss, and packet loss is assumed upon time out. The draft allows to control the number of outstanding fragments, that have been transmitted but for which an acknowledgment was not received yet. It must be noted that the number of outstanding fragments should not exceed the number of hops in the network, but the way to figure the number of hops is out of scope for this document.

Congestion on the forward path can also be indicated by an Explicit Congestion Notification (ECN) mechanism. Though whether and how ECN [RFC3168] is carried out over the LoWPAN is out of scope, this draft provides a way for the destination endpoint to echo an ECN indication back to the source endpoint in an acknowledgment message as represented in Figure 5 in Section 4.2.

It must be noted that congestion and collision are different topics. In particular, when a mesh operates on a same channel over multiple hops, then the forwarding of a fragment over a certain hop may collide with the forwarding of a next fragment that is following over a previous hop but in a same interference domain. This draft enables an end-to-end flow control, but leaves it to the sender stack to pace individual fragments within a transmit window, so that a given fragment is sent only when the previous fragment has had a chance to progress beyond the interference domain of this hop. In the case of 6TiSCH [I-D.ietf-6tisch-architecture], which operates over the TimeSlotted Channel Hopping [RFC7554] (TSCH) mode of operation of IEEE802.14.5, a fragment is forwarded over a different channel at a different time and it makes full sense to transmit the next fragment as soon as the previous fragment has had its chance to be forwarded at the next hop.

From the standpoint of a source 6LoWPAN endpoint, an outstanding fragment is a fragment that was sent but for which no explicit acknowledgment was received yet. This means that the fragment might be on the way, received but not yet acknowledged, or the

acknowledgment might be on the way back. It is also possible that either the fragment or the acknowledgment was lost on the way.

From the sender standpoint, all outstanding fragments might still be in the network and contribute to its congestion. There is an assumption, though, that after a certain amount of time, a frame is either received or lost, so it is not causing congestion anymore. This amount of time can be estimated based on the round trip delay between the 6LoWPAN endpoints. The method detailed in [RFC6298] is recommended for that computation.

The reader is encouraged to read through "Congestion Control Principles" [RFC2914]. Additionally [RFC7567] and [RFC5681] provide deeper information on why this mechanism is needed and how TCP handles Congestion Control. Basically, the goal here is to manage the amount of fragments present in the network; this is achieved by to reducing the number of outstanding fragments over a congested path by throttling the sources.

Section 5 describes how the sender decides how many fragments are (re)sent before an acknowledgment is required, and how the sender adapts that number to the network conditions.

#### Authors' Addresses

Pascal Thubert (editor)  
Cisco Systems, Inc  
Building D  
45 Allée des Ormes - BP1200  
MOUGINS - Sophia Antipolis 06254  
FRANCE

Phone: +33 497 23 26 34  
Email: pthubert@cisco.com

Jonathan W. Hui  
Nest Labs  
3400 Hillview Ave  
Palo Alto, California 94304  
USA

Email: jonhui@nestlabs.com