Network Working Group
Request for Comments: 4944
Category: Standards Track

G. Montenegro
Microsoft Corporation
N. Kushalnagar
Intel Corp
J. Hui
D. Culler
Arch Rock Corp

September 2007

Transmission of IPv6 Packets over IEEE 802.15.4 Networks

Status of This Memo

Abstract

This document describes the frame format for transmission of IPv6 packets and the method of forming IPv6 link-local addresses and statelessly autoconfigured addresses on IEEE 802.15.4 networks. Additional specifications include a simple header compression scheme using shared context and provisions for packet delivery in IEEE 802.15.4 meshes.

Table of Contents

1. Introduction

The IEEE 802.15.4 standard [ieee802.15.4] targets low-power personal area networks. 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 addresses on top of IEEE 802.15.4 networks. Since IPv6 requires support of packet sizes much larger than the largest IEEE 802.15.4 frame size, an adaptation layer is defined. This document also defines mechanisms for header compression required to make IPv6 practical on IEEE 802.15.4 networks, and the provisions required for packet delivery in IEEE 802.15.4 meshes. However, a full specification of mesh routing (the specific protocol used, the interactions with neighbor discovery, etc) is out of the scope of this document.

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

AES: Advanced Encryption Scheme

CSMA/CA: Carrier Sense Multiple Access / Collision Avoidance

FFD: Full Function Device

GTS: Guaranteed Time Service

MTU: Maximum Transmission Unit

MAC: Media Access Control

PAN: Personal Area Network

RFD: Reduced Function Device

2. IEEE 802.15.4 Mode for IP

IEEE 802.15.4 defines four types of frames: beacon frames, MAC command frames, acknowledgement frames, and data frames. IPv6 packets MUST be carried on data frames. Data frames may optionally request that they be acknowledged. In keeping with [RFC3819], it is recommended that IPv6 packets be carried in frames for which acknowledgements are requested so as to aid link-layer recovery.

IEEE 802.15.4 networks can either be nonbeacon-enabled or beacon-enabled [ieee802.15.4]. The latter is an optional mode in which devices are synchronized by a so-called coordinator's beacons. This allows the use of superframes within which a contention-free Guaranteed Time Service (GTS) is possible. This document does not require that IEEE networks run in beacon-enabled mode. In nonbeacon-enabled networks, data frames (including those carrying IPv6 packets) are sent via the contention-based channel access method of unslotted CSMA/CA.

In nonbeacon-enabled networks, beacons are not used for synchronization. However, they are still useful for link-layer device discovery to aid in association and disassociation events. This document recommends that beacons be configured so as to aid these functions. A further recommendation is for these events to be

available at the IPv6 layer to aid in detecting network attachment, a problem being worked on at the IETF at the time of this writing.

The specification allows for frames in which either the source or destination addresses (or both) are elided. The mechanisms defined in this document require that both source and destination addresses be included in the IEEE 802.15.4 frame header. The source or destination PAN ID fields may also be included.

3. Addressing Modes

IEEE 802.15.4 defines several addressing modes: it allows the use of either IEEE 64-bit extended addresses or (after an association event) 16-bit addresses unique within the PAN [ieee802.15.4]. This document supports both 64-bit extended addresses, and 16-bit short addresses. For use within 6LoWPANs, this document imposes additional constraints (beyond those imposed by IEEE 802.15.4) on the format of the 16-bit short addresses, as specified in Section 12. Short addresses being transient in nature, a word of caution is in order: since they are doled out by the PAN coordinator function during an association event, their validity and uniqueness is limited by the lifetime of that association. This can be cut short by the expiration of the association or simply by any mishap occurring to the PAN coordinator. Because of the scalability issues posed by such a centralized allocation and single point of failure at the PAN coordinator, deployers should carefully weigh the tradeoffs (and implement the necessary mechanisms) of growing such networks based on short addresses. Of course, IEEE 64-bit extended addresses may not suffer from these drawbacks, but still share the remaining scalability issues concerning routing, discovery, configuration, etc.

This document assumes that a PAN maps to a specific IPv6 link. This complies with the recommendation that shared networks support link-layer subnet [RFC3819] broadcast. Strictly speaking, it is multicast not broadcast that exists in IPv6. However, multicast is not supported natively in IEEE 802.15.4. Hence, IPv6 level multicast packets MUST be carried as link-layer broadcast frames in IEEE 802.15.4 networks. This MUST be done such that the broadcast frames are only heeded by devices within the specific PAN of the link in question. As per Section 7.5.6.2 in [ieee802.15.4], this is accomplished as follows:

1. A destination PAN identifier is included in the frame, and it MUST match the PAN ID of the link in question.

2. A short destination address is included in the frame, and it MUST match the broadcast address (0xffff).

Additionally, support for mapping of IPv6 multicast addresses per Section 9 MUST only be used in a mesh configuration. A full specification of such functionality is out of the scope of this document.

As usual, hosts learn IPv6 prefixes via router advertisements as per [RFC4861].

4. Maximum Transmission Unit

The MTU size for IPv6 packets over IEEE 802.15.4 is 1280 octets. However, a full IPv6 packet does not fit in an IEEE 802.15.4 frame. 802.15.4 protocol data units have different sizes depending on how much overhead is present [ieee802.15.4]. Starting from a maximum physical layer packet size of 127 octets (aMaxPHYPacketSize) and a maximum frame overhead of 25 (aMaxFrameOverhead), the resultant maximum frame size at the media access control layer is 102 octets. Link-layer security imposes further overhead, which in the maximum case (21 octets of overhead in the AES-CCM-128 case, versus 9 and 13 for AES-CCM-32 and AES-CCM-64, respectively) leaves only 81 octets available. This is obviously far below the minimum IPv6 packet size of 1280 octets, and in keeping with Section 5 of the IPv6 specification [RFC2460], a fragmentation and reassembly adaptation layer must be provided at the layer below IP. Such a layer is defined below in Section 5.

Furthermore, since the IPv6 header is 40 octets long, this leaves only 41 octets for upper-layer protocols, like UDP. The latter uses 8 octets in the header which leaves only 33 octets for application data. Additionally, as pointed out above, there is a need for a fragmentation and reassembly layer, which will use even more octets.

The above considerations lead to the following two observations:

1. The adaptation layer must be provided to comply with the IPv6 requirements of a minimum MTU. However, it is expected that (a) most applications of IEEE 802.15.4 will not use such large packets, and (b) small application payloads in conjunction with the proper header compression will produce packets that fit within a single IEEE 802.15.4 frame. The justification for this adaptation layer is not just for IPv6 compliance, as it is quite likely that the packet sizes produced by certain application exchanges (e.g., configuration or provisioning) may require a small number of fragments.

2. Even though the above space calculation shows the worst-case scenario, it does point out the fact that header compression is compelling to the point of almost being unavoidable. Since we

expect that most (if not all) applications of IP over IEEE 802.15.4 will make use of header compression, it is defined below in Section 10.

5. LoWPAN Adaptation Layer and Frame Format

The encapsulation formats defined in this section (subsequently referred to as the "LoWPAN encapsulation") are the payload in the IEEE 802.15.4 MAC protocol data unit (PDU). The LoWPAN payload (e.g., an IPv6 packet) follows this encapsulation header.

All LoWPAN encapsulated datagrams transported over IEEE 802.15.4 are prefixed by an encapsulation header stack. Each header in the header stack contains 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 LoWPAN header, the analogous header sequence is mesh (L2) addressing, hop-by-hop options (including L2 broadcast/multicast), fragmentation, and finally payload. These examples show typical header stacks that may be used in a LoWPAN network.

A LoWPAN encapsulated IPv6 datagram:

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

A LoWPAN encapsulated LOWPAN_HC1 compressed IPv6 datagram:

```
+--------------+------------+---------+
| HC1 Dispatch | HC1 Header | Payload |
+--------------+------------+---------+
```

A LoWPAN encapsulated LOWPAN_HC1 compressed IPv6 datagram that requires mesh addressing:

```
+-----------+-------------+--------------+------------+---------+
| Mesh Type | Mesh Header | HC1 Dispatch | HC1 Header | Payload |
+-----------+-------------+--------------+------------+---------+
```

A LoWPAN encapsulated LOWPAN_HC1 compressed IPv6 datagram that requires fragmentation:

```
+-----------+-------------+--------------+------------+---------+
| Frag Type | Frag Header | HC1 Dispatch | HC1 Header | Payload |
+-----------+-------------+--------------+------------+---------+
```

A LoWPAN encapsulated LOWPAN_HC1 compressed IPv6 datagram that requires both mesh addressing and fragmentation:

```
+-------+-------+-------+-------+---------+---------+---------+
| M Typ | M Hdr | F Typ | F Hdr | HC1 Dsp | HC1 Hdr | Payload |
+-------+-------+-------+-------+---------+---------+---------+
```

A LoWPAN encapsulated LOWPAN_HC1 compressed IPv6 datagram that requires both mesh addressing and a broadcast header to support mesh broadcast/multicast:

```
+-------+-------+-------+-------+---------+---------+---------+
| M Typ | M Hdr | B Dsp | B Hdr | HC1 Dsp | HC1 Hdr | Payload |
+-------+-------+-------+-------+---------+---------+---------+
```

When more than one LoWPAN header is used in the same packet, they MUST appear in the following order:

Mesh Addressing Header

Broadcast Header

Fragmentation Header

All protocol datagrams (e.g., IPv6, compressed IPv6 headers, etc.) SHALL be preceded by one of the valid LoWPAN encapsulation headers, examples of which are given above. This permits uniform software treatment of datagrams without regard to the mode of their transmission.

The definition of LoWPAN headers, other than mesh addressing and fragmentation, 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 LoWPAN adaptation layer, it is not intended to provided 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 Type and Header

The dispatch type is defined by a zero bit as the first bit and a one bit as the second bit. The dispatch type and header are shown here:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 1| Dispatch  |  type-specific header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

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

Figure 1: Dispatch Type and Header

The dispatch value may be treated as an unstructured namespace. Only a few symbols are required to represent current LoWPAN 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.

Pattern	Header Type
00 xxxxxx	NALP - Not a LoWPAN frame
01 000001	IPv6 - Uncompressed IPv6 Addresses
01 000010	LOWPAN_HC1 - LOWPAN_HC1 compressed IPv6
01 000011	reserved - Reserved for future use
...	reserved - Reserved for future use
01 001111	reserved - Reserved for future use
01 010000	LOWPAN_BC0 - LOWPAN_BC0 broadcast
01 010001	reserved - Reserved for future use
...	reserved - Reserved for future use
01 111110	reserved - Reserved for future use
01 111111	ESC - Additional Dispatch byte follows
10 xxxxxx	MESH - Mesh Header
11 000xxx	FRAG1 - Fragmentation Header (first)
11 001000	reserved - Reserved for future use
...	reserved - Reserved for future use
11 011111	reserved - Reserved for future use
11 100xxx	FRAGN - Fragmentation Header (subsequent)
11 101000	reserved - Reserved for future use
...	reserved - Reserved for future use
11 111111	reserved - Reserved for future use

Figure 2: Dispatch Value Bit Pattern

NALP: Specifies that the following bits are not a part of the LoWPAN encapsulation, and any LoWPAN node that encounters a dispatch value of 00xxxxxx shall discard the packet. Other non-LoWPAN protocols that wish to coexist with LoWPAN nodes should include a byte matching this pattern immediately following the 802.15.4. header.

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

LOWPAN_HC1: Specifies that the following header is a LOWPAN_HC1 compressed IPv6 header. This header format is defined in Figure 9.

LOWPAN_BC0: Specifies that the following header is a LOWPAN_BC0 header for mesh broadcast/multicast support and is described in Section 11.1.

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.

5.2. Mesh Addressing Type and Header

The mesh type is defined by a one bit and zero bit as the first two bits. The mesh type and header are shown here:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|V|F|HopsLft| originator address, final address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Mesh Addressing Type and Header

Field definitions are as follows:

V: This 1-bit field SHALL be zero if the Originator (or "Very first") Address is an IEEE extended 64-bit address (EUI-64), or 1 if it is a short 16-bit addresses.

F: This 1-bit field SHALL be zero if the Final Destination Address is an IEEE extended 64-bit address (EUI-64), or 1 if it is a short 16-bit addresses.

Hops Left: This 4-bit field SHALL be decremented by each forwarding node before sending this packet towards its next hop. The packet is not forwarded any further if Hops Left is decremented to zero. The value 0xF is reserved and signifies an 8-bit Deep Hops Left field immediately following, and allows a source node to specify a hop limit greater than 14 hops.

Originator Address: This is the link-layer address of the Originator.

Final Destination Address: This is the link-layer address of the Final Destination.

Note that the 'V' and 'F' bits allow for a mix of 16 and 64-bit addresses. This is useful at least to allow for mesh layer "broadcast", as 802.15.4 broadcast addresses are defined as 16-bit short addresses.

A further discussion of frame delivery within a mesh is in Section 11.

5.3. Fragmentation Type and Header

If an entire payload (e.g., IPv6) datagram fits within a single 802.15.4 frame, it is unfragmented and the LoWPAN encapsulation should not contain a fragmentation header. If the datagram does not fit within a single IEEE 802.15.4 frame, it SHALL be broken into link fragments. As the fragment offset can only express multiples of eight bytes, all link fragments for a datagram except the last one MUST be multiples of eight bytes in length. The first link fragment SHALL contain the first fragment header as defined below.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 0 0 0|    datagram_size    |         datagram_tag          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: First Fragment

The second and subsequent link fragments (up to and including the last) SHALL contain a fragmentation header that conforms to the format shown below.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 0 0|    datagram_size    |         datagram_tag          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|datagram_offset|
+-+-+-+-+-+-+-+-+
```

Figure 5: Subsequent Fragments

datagram_size: This 11-bit field encodes the size of the entire IP packet before link-layer fragmentation (but after IP layer fragmentation). The value of datagram_size SHALL be the same for all link-layer fragments of an IP packet. For IPv6, this SHALL be 40 octets (the size of the uncompressed IPv6 header) more than the value of Payload Length in the IPv6 header [RFC2460] of the packet. Note that this packet may already be fragmented by hosts involved in the communication, i.e., this field needs to encode a maximum length of 1280 octets (the IEEE 802.15.4 link MTU, as defined in this document).

NOTE: This field does not need to be in every packet, as one could send it with the first fragment and elide it subsequently. However, including it in every link fragment eases the task of reassembly in the event that a second (or subsequent) link

fragment arrives before the first. In this case, the guarantee of learning the datagram_size as soon as any of the fragments arrives tells the receiver how much buffer space to set aside as it waits for the rest of the fragments. The format above trades off simplicity for efficiency.

datagram_tag: The value of datagram_tag (datagram tag) SHALL be the same for all link fragments of a payload (e.g., IPv6) datagram. The sender SHALL increment datagram_tag for successive, fragmented datagrams. The incremented value of datagram_tag SHALL wrap from 65535 back to zero. This field is 16 bits long, and its initial value is not defined.

datagram_offset: This field is present only in the second and subsequent link fragments and SHALL specify the offset, in increments of 8 octets, of the fragment from the beginning of the payload datagram. The first octet of the datagram (e.g., the start of the IPv6 header) has an offset of zero; the implicit value of datagram_offset in the first link fragment is zero. This field is 8 bits long.

The recipient of link fragments SHALL use (1) the sender's 802.15.4 source address (or the Originator Address if a Mesh Addressing field is present), (2) the destination's 802.15.4 address (or the Final Destination address if a Mesh Addressing field is present), (3) datagram_size, and (4) datagram_tag to identify all the link fragments that belong to a given datagram.

Upon receipt of a link fragment, the recipient starts constructing the original unfragmented packet whose size is datagram_size. It uses the datagram_offset field to determine the location of the individual fragments within the original unfragmented packet. For example, it may place the data payload (except the encapsulation header) within a payload datagram reassembly buffer at the location specified by datagram_offset. The size of the reassembly buffer SHALL be determined from datagram_size.

If a link fragment that overlaps another fragment is received, as identified above, and differs in either the size or datagram_offset of the overlapped fragment, the fragment(s) already accumulated in the reassembly buffer SHALL be discarded. A fresh reassembly may be commenced with the most recently received link fragment. Fragment overlap is determined by the combination of datagram_offset from the encapsulation header and "Frame Length" from the 802.15.4 Physical Layer Protocol Data Unit (PPDU) packet header.

Upon detection of a IEEE 802.15.4 Disassociation event, fragment recipients MUST discard all link fragments of all partially

reassembled payload datagrams, and fragment senders MUST discard all not yet transmitted link fragments of all partially transmitted payload (e.g., IPv6) datagrams. Similarly, when a node first receives a fragment with a given datagram_tag, it starts a reassembly timer. When this time expires, if the entire packet has not been reassembled, the existing fragments MUST be discarded and the reassembly state MUST be flushed. The reassembly timeout MUST be set to a maximum of 60 seconds (this is also the timeout in the IPv6 reassembly procedure [RFC2460]).

6. Stateless Address Autoconfiguration

This section defines how to obtain an IPv6 interface identifier.

The Interface Identifier [RFC4291] for an IEEE 802.15.4 interface may be based on the EUI-64 identifier [EUI64] assigned to the IEEE 802.15.4 device. In this case, the Interface Identifier is formed from the EUI-64 according to the "IPv6 over Ethernet" specification [RFC2464].

All 802.15.4 devices have an IEEE EUI-64 address, but 16-bit short addresses (Section 3 and Section 12) are also possible. In these cases, a "pseudo 48-bit address" is formed as follows. First, the left-most 32 bits are formed by concatenating 16 zero bits to the 16-bit PAN ID (alternatively, if no PAN ID is known, 16 zero bits may be used). This produces a 32-bit field as follows:

```
16_bit_PAN:16_zero_bits
```

Then, these 32 bits are concatenated with the 16-bit short address. This produces a 48-bit address as follows:

```
32_bits_as_specified_previously:16_bit_short_address
```

The interface identifier is formed from this 48-bit address as per the "IPv6 over Ethernet" specification [RFC2464]. However, in the resultant interface identifier, the "Universal/Local" (U/L) bit SHALL be set to zero in keeping with the fact that this is not a globally unique value. For either address format, all zero addresses MUST NOT be used.

A different MAC address set manually or by software MAY be used to derive the Interface Identifier. If such a MAC address is used, its global uniqueness property should be reflected in the value of the U/L bit.

An IPv6 address prefix used for stateless autoconfiguration [RFC4862] of an IEEE 802.15.4 interface MUST have a length of 64 bits.

7. IPv6 Link Local Address

The IPv6 link-local address [RFC4291] for an IEEE 802.15.4 interface is formed by appending the Interface Identifier, as defined above, to the prefix FE80::/64.

```
   10 bits            54 bits                  64 bits
+----------+-----------------------+----------------------------+
|1111111010|         (zeros)       |    Interface Identifier    |
+----------+-----------------------+----------------------------+
```

Figure 6

8. Unicast Address Mapping

The address resolution procedure for mapping IPv6 non-multicast addresses into IEEE 802.15.4 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 forms when the link layer is IEEE 802.15.4 and the addresses are EUI-64 or 16-bit short addresses, respectively.

```
                       0                   1
                       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |     Type      |    Length=2   |
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |                               |
                      +-         IEEE 802.15.4       -+
                      |          EUI-64               |
                      +-                             -+
                      |                               |
                      +-          Address            -+
                      |                               |
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |                               |
                      +-          Padding            -+
                      |                               |
                      +-         (all zeros)         -+
                      |                               |
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                       0                   1
                       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |     Type      |    Length=1   |
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |     16-bit short Address      |
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |                               |
                      +-          Padding            -+
                      |          (all zeros)          |
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7

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 2 if using EUI-64 addresses, or 1 if using 16-bit short addresses.

IEEE 802.15.4 Address: The 64-bit IEEE 802.15.4 address, or the 16-bit short address (as per the format in Section 9), in canonical

bit order. This is the address the interface currently responds to. This address may be different from the built-in address used to derive the Interface Identifier, because of privacy or security (e.g., of neighbor discovery) considerations.

9. Multicast Address Mapping

The functionality in this section MUST only be used in a mesh-enabled LoWPAN. An IPv6 packet with a multicast destination address (DST), consisting of the sixteen octets DST[1] through DST[16], is transmitted to the following 802.15.4 16-bit multicast address:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 0|DST[15]* |   DST[16]     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8

Here, DST[15]* refers to the last 5 bits in octet DST[15], that is, bits 3-7 within DST[15]. The initial 3-bit pattern of "100" follows the 16-bit address format for multicast addresses (Section 12).

This allows for multicast support within 6LoWPAN networks, but the full specification of such support is out of the scope of this document. Example mechanisms are: flooding, controlled flooding, unicasting to the PAN coordinator, etc. It is expected that this would be specified by the different mesh routing mechanisms.

10. Header Compression

There is much published and in-progress standardization work on header compression. Nevertheless, header compression for IPv6 over IEEE 802.15.4 has differing constraints summarized as follows:

Existing work assumes that there are many flows between any two devices. Here, we assume a very simple and low-context flavor of header compression. Whereas this works independently of flows (potentially several), it does not use any context specific to any flow. Thus, it cannot achieve as much compression as schemes that build a separate context for each flow to be compressed.

Given the very limited packet sizes, it is highly desirable to integrate layer 2 with layer 3 compression, something traditionally not done (although now changing due to the ROHC (RObust Header Compression) working group).

It is expected that IEEE 802.15.4 devices will be deployed in multi-hop networks. However, header compression in a mesh departs from the usual point-to-point link scenario in which the compressor and decompressor are in direct and exclusive communication with each other. In an IEEE 802.15.4 network, it is highly desirable for a device to be able to send header compressed packets via any of its neighbors, with as little preliminary context-building as possible.

Any new packet formats required by header compression reuse the basic packet formats defined in Section 5 by using different dispatch values.

Header compression may result in alignment not falling on an octet boundary. Since hardware typically cannot transmit data in units less than an octet, padding must be used. Padding is done as follows: First, the entire series of contiguous compressed headers is laid out (this document only defines IPv6 and UDP header compression schemes, but others may be defined elsewhere). Then, zero bits SHOULD be added as appropriate to align to an octet boundary. This counteracts any potential misalignment caused by header compression, so subsequent fields (e.g., non-compressed headers or data payloads) start on an octet boundary and follow as usual.

10.1. Encoding of IPv6 Header Fields

By virtue of having joined the same 6LoWPAN network, devices share some state. This makes it possible to compress headers without explicitly building any compression context state. Therefore, 6LoWPAN header compression does not keep any flow state; instead, it relies on information pertaining to the entire link. The following IPv6 header values are expected to be common on 6LoWPAN networks, so the HC1 header has been constructed to efficiently compress them from the onset: Version is IPv6; both IPv6 source and destination addresses are link local; the IPv6 interface identifiers (bottom 64 bits) for the source or destination addresses can be inferred from the layer two source and destination addresses (of course, this is only possible for interface identifiers derived from an underlying 802.15.4 MAC address); the packet length can be inferred either from layer two ("Frame Length" in the IEEE 802.15.4 PPDU) or from the "datagram_size" field in the fragment header (if present); both the Traffic Class and the Flow Label are zero; and the Next Header is UDP, ICMP or TCP. The only field in the IPv6 header that always needs to be carried in full is the Hop Limit (8 bits). Depending on how closely the packet matches this common case, different fields may not be compressible thus needing to be carried "in-line" as well (Section 10.3.1). This common IPv6 header (as mentioned above) can be compressed to 2 octets (1 octet for the HC1 encoding and 1 octet

for the Hop Limit), instead of 40 octets. Such a packet is compressible via the LOWPAN_HC1 format by using a Dispatch value of LOWPAN_HC1 followed by a LOWPAN_HC1 header "HC1 encoding" field (8 bits) to encode the different combinations as shown below. This header may be preceded by a fragmentation header, which may be preceded by a mesh header.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| HC1 encoding  |     Non-Compressed fields follow...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: LOWPAN_HC1 (common compressed header encoding)

As can be seen below (bit 7), an HC2 encoding may follow an HC1 octet. In this case, the non-compressed fields follow the HC2 encoding field (Section 10.3).

The address fields encoded by "HC1 encoding" are interpreted as follows:

PI: Prefix carried in-line (Section 10.3.1).

PC: Prefix compressed (link-local prefix assumed).

II: Interface identifier carried in-line (Section 10.3.1).

IC: Interface identifier elided (derivable from the corresponding link-layer address). If applied to the interface identifier of either the source or destination address when routing in a mesh (Section 11), the corresponding link-layer address is that found in the "Mesh Addressing" field (Section 5.2).

The "HC1 encoding" is shown below (starting with bit 0 and ending at bit 7):

IPv6 source address (bits 0 and 1):

00: PI, II

01: PI, IC

10: PC, II

11: PC, IC

IPv6 destination address (bits 2 and 3):

00: PI, II

01: PI, IC

10: PC, II

11: PC, IC

Traffic Class and Flow Label (bit 4):

0: not compressed; full 8 bits for Traffic Class and 20 bits for Flow Label are sent

1: Traffic Class and Flow Label are zero

Next Header (bits 5 and 6):

00: not compressed; full 8 bits are sent

01: UDP

10: ICMP

11: TCP

HC2 encoding(bit 7):

0: No more header compression bits

1: HC1 encoding immediately followed by more header compression bits per HC2 encoding format. Bits 5 and 6 determine which of the possible HC2 encodings apply (e.g., UDP, ICMP, or TCP encodings).

10.2. Encoding of UDP Header Fields

Bits 5 and 6 of the LOWPAN_HC1 allows compressing the Next Header field in the IPv6 header (for UDP, TCP, and ICMP). Further compression of each of these protocol headers is also possible. This section explains how the UDP header itself may be compressed. The HC2 encoding in this section is the HC_UDP encoding, and it only applies if bits 5 and 6 in HC1 indicate that the protocol that follows the IPv6 header is UDP. The HC_UDP encoding (Figure 10) allows compressing the following fields in the UDP header: source port, destination port, and length. The UDP header's checksum field is not compressed and is therefore carried in full. The scheme

defined below allows compressing the UDP header to 4 octets instead of the original 8 octets.

The only UDP header field whose value may be deduced from information available elsewhere is the Length. The other fields must be carried in-line either in full or in a partially compressed manner (Section 10.3.2).

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|HC_UDP encoding|     Fields carried in-line follow...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: HC_UDP (UDP common compressed header encoding)

The "HC_UDP encoding" for UDP is shown below (starting with bit 0 and ending at bit 7):

UDP source port (bit 0):

0: Not compressed, carried "in-line" (Section 10.3.2)

1: Compressed to 4 bits. The actual 16-bit source port is obtained by calculating: P + short_port value. The value of P is the number 61616 (0xF0B0). The short_port is expressed as a 4-bit value which is carried "in-line" (Section 10.3.2)

UDP destination port (bit 1):

0: Not compressed, carried "in-line" (Section 10.3.2)

1: Compressed to 4 bits. The actual 16-bit destination port is obtained by calculating: P + short_port value. The value of P is the number 61616 (0xF0B0). The short_port is expressed as a 4-bit value which is carried "in-line" (Section 10.3.2)

Length (bit 2):

0: not compressed, carried "in-line" (Section 10.3.2)

1: compressed, length computed from IPv6 header length information. The value of the UDP length field is equal to the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the UDP header.

Reserved (bit 3 through 7)

10.3. Non-Compressed Fields

10.3.1. Non-Compressed IPv6 Fields

This scheme allows the IPv6 header to be compressed to different degrees. Hence, instead of the entire (standard) IPv6 header, only non-compressed fields need to be sent. The subsequent header (as specified by the Next Header field in the original IPv6 header) immediately follows the IPv6 non-compressed fields.

Uncompressed IPv6 addressing is described by a dispatch type containing an IPv6 dispatch value followed by the uncompressed IPv6 header. This dispatch type may be preceded by additional LoWPAN headers.

The non-compressed IPv6 field that MUST be always present is the Hop Limit (8 bits). This field MUST always follow the encoding fields (e.g., "HC1 encoding" as shown in Figure 9), perhaps including other future encoding fields). Other non-compressed fields MUST follow the Hop Limit as implied by the "HC1 encoding" in the exact same order as shown above (Section 10.1): source address prefix (64 bits) and/or interface identifier (64 bits), destination address prefix (64 bits) and/or interface identifier (64 bits), Traffic Class (8 bits), Flow Label (20 bits) and Next Header (8 bits). The actual next header (e.g., UDP, TCP, ICMP, etc) follows the non-compressed fields.

10.3.2. Non-Compressed and Partially Compressed UDP Fields

This scheme allows the UDP header to be compressed to different degrees. Hence, instead of the entire (standard) UDP header, only non-compressed or partially compressed fields need to be sent.

The non-compressed or partially compressed fields in the UDP header MUST always follow the IPv6 header and any of its associated in-line fields. Any UDP header in-line fields present MUST appear in the same order as the corresponding fields appear in a normal UDP header [RFC0768], e.g., source port, destination port, length, and checksum. If either the source or destination ports are in "short_port" notation (as indicated in the compressed UDP header), then instead of taking 16 bits, the inline port numbers take 4 bits.

11. Frame Delivery in a Link-Layer Mesh

Even though 802.15.4 networks are expected to commonly use mesh routing, the IEEE 802.15.4-2003 specification [ieee802.15.4] does not define such capability. In such cases, Full Function Devices (FFDs) run an ad hoc or mesh routing protocol to populate their routing tables (outside the scope of this document). In such mesh scenarios,

two devices do not require direct reachability in order to communicate. Of these devices, the sender is known as the "Originator", and the receiver is known as the "Final Destination". An originator device may use other intermediate devices as forwarders towards the final destination. In order to achieve such frame delivery using unicast, it is necessary to include the link-layer addresses of the originator and final destinations, in addition to the hop-by-hop source and destination.

This section defines how to effect delivery of layer 2 frames in a mesh, given a target "Final Destination" link-layer address.

Mesh delivery is enabled by including a Mesh Addressing header prior to any other headers of the LoWPAN encapsulation (Section 5), an unfragmented and fragmented header; a full-blown IPv6 header; or a compressed IPv6 header as per Section 10 or any others defined elsewhere.

If a node wishes to use a default mesh forwarder to deliver a packet (i.e., because it does not have direct reachability to the destination), it MUST include a Mesh Addressing header with the originator's link-layer address set to its own, and the final destination's link-layer address set to the packet's ultimate destination. It sets the source address in the 802.15.4 header to its own link-layer address, and puts the forwarder's link-layer address in the 802.15.4 header's destination address field. Finally, it transmits the packet.

Similarly, if a node receives a frame with a Mesh Addressing header, it must look at the Mesh Addressing header's "Final Destination" field to determine the real destination. If the node is itself the final destination, it consumes the packet as per normal delivery. If it is not the final destination, the device then reduces the "Hops Left" field, and if the result is zero, discards the packet. Otherwise, the node consults its link-layer routing table, determines what the next hop towards the final destination should be, and puts that address in the destination address field of the 802.15.4 header. Finally, the node changes the source address in the 802.15.4 header to its own link-layer address and transmits the packet.

Whereas a node must participate in a mesh routing protocol to be a forwarder, no such requirement exists for simply using mesh forwarding. Only "Full Function Devices" (FFDs) are expected to participate as routers in a mesh. "Reduced Function Devices" (RFDs) limit themselves to discovering FFDs and using them for all their forwarding, in a manner similar to how IP hosts typically use default routers to forward all their off-link traffic. For an RFD using mesh delivery, the "forwarder" is always the appropriate FFD.

11.1. LoWPAN Broadcast

Additional mesh routing functionality is encoded using a routing header immediately following the Mesh header. In particular, a broadcast header consists of a LOWPAN_BC0 dispatch followed by a sequence number field. The sequence number is used to detect duplicate packets (and hopefully suppress them).

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1|LOWPAN_BC0 |Sequence Number|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Broadcast Header

Field definitions are as follows:

Sequence Number: This 8-bit field SHALL be incremented by the originator whenever it sends a new mesh broadcast or multicast packet. Full specification of how to handle this field is out of the scope of this document.

Further implications of such mesh-layer broadcast, e.g., whether it maps to a controlled flooding mechanism or its role in, say, topology discovery, is out of the scope of this document.

Additional mesh routing capabilities, such as specifying the mesh routing protocol, source routing, and so on may be expressed by defining additional routing headers that precede the fragmentation or addressing header in the header stack. The full specification of such mesh routing capabilities are out of the scope of this document.

12. IANA Considerations

This document creates two new IANA registries, as discussed below. Future assignments in these registries are to be coordinated via IANA under the policy of "Specification Required" [RFC2434]. It is expected that this policy will allow for other (non-IETF) organizations to more easily obtain assignments.

This document creates a new IANA registry for the Dispatch type field shown in the header definitions in Section 5. This document defines the values IPv6, LOWPAN_HC1 header compression, BC0 broadcast and two escape patterns (NALP to indicate not a LOWPAN frame and ESC to allow additional dispatch bytes). This document defines this field to be 8 bits long. The values 00xxxxxx being reserved and not used, allows for a total of 192 different values, which should be more than

enough. If header compression formats in addition to HC1 are defined or if additional TCP, ICMP HC2 formats are defined, it is expected that these will use reserved dispatch values following LOWPAN_HC1. If additional mesh delivery formats are defined these will use reserved values following LOWPAN_BC0.

This document creates a new IANA registry for the 16-bit short address fields as used in 6LoWPAN packets.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     16-bit short Address      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12

This registry MUST include the addresses 0xffff (16-bit broadcast address accepted by all devices currently listening to the channel) and 0xfffe as defined in [ieee802.15.4]. Additionally, within 6LoWPAN networks, 16-bit short addresses MUST follow this format (referring to bit fields in the order from 0 to 7), where "x" is a place holder for an unspecified bit value:

Range 1, 0xxxxxxxxxxxxxxx: The first bit (bit 0) SHALL be zero if the 16-bit address is a unicast address. This leaves 15 bits for the actual address.

Range 2, 100xxxxxxxxxxxxx: Bits 0, 1, and 2 SHALL follow this pattern if the 16-bit address is a multicast address (see Section 9). This leaves 13 bits for the actual multicast address.

Range 3, 101xxxxxxxxxxxxx: This pattern for bits 0, 1, and 2 is reserved. Any future assignment shall follow the policy mentioned above.

Range 4, 110xxxxxxxxxxxxx: This pattern for bits 0, 1, and 2 is reserved. Any future assignment shall follow the policy mentioned above.

Range 5, 111xxxxxxxxxxxxx: This pattern for bits 0, 1, and 2 is reserved. Any future assignment shall follow the policy mentioned above.

13. Security Considerations

The method of derivation of Interface Identifiers from EUI-64 MAC addresses is intended to preserve global uniqueness when possible. However, there is no protection from duplication through accident or forgery.

Neighbor Discovery in IEEE 802.15.4 links may be susceptible to threats as detailed in [RFC3756]. Mesh routing is expected to be common in IEEE 802.15.4 networks. This implies additional threats due to ad hoc routing as per [KW03]. IEEE 802.15.4 provides some capability for link-layer security. Users are urged to make use of such provisions if at all possible and practical. Doing so will alleviate the threats stated above.

A sizeable portion of IEEE 802.15.4 devices is expected to always communicate within their PAN (i.e., within their link, in IPv6 terms). In response to cost and power consumption considerations, and in keeping with the IEEE 802.15.4 model of "Reduced Function Devices" (RFDs), these devices will typically implement the minimum set of features necessary. Accordingly, security for such devices may rely quite strongly on the mechanisms defined at the link layer by IEEE 802.15.4. The latter, however, only defines the Advanced Encryption Standard (AES) modes for authentication or encryption of IEEE 802.15.4 frames, and does not, in particular, specify key management (presumably group oriented). Other issues to address in real deployments relate to secure configuration and management. Whereas such a complete picture is out of the scope of this document, it is imperative that IEEE 802.15.4 networks be deployed with such considerations in mind. Of course, it is also expected that some IEEE 802.15.4 devices (the so-called "Full Function Devices", or "FFDs") will implement coordination or integration functions. These may communicate regularly with off-link IPv6 peers (in addition to the more common on-link exchanges). Such IPv6 devices are expected to secure their end-to-end communications with the usual mechanisms (e.g., IPsec, TLS, etc).

14. Acknowledgements

Thanks to the authors of RFC 2464 and RFC 2734, as parts of this document are patterned after theirs. Thanks to Geoff Mulligan for useful discussions which helped shape this document. Erik Nordmark's suggestions were instrumental for the header compression section. Also thanks to Shoichi Sakane, Samita Chakrabarti, Vipul Gupta, Carsten Bormann, Ki-Hyung Kim, Mario Mao, Phil Levis, Magnus Westerlund, and Jari Arkko.

15. References

15.1. Normative References

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

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[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.

[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.

[ieee802.15.4] IEEE Computer Society, "IEEE Std. 802.15.4-2003", October 2003.

15.2. Informative References

[EUI64] "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", IEEE http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

[KW03] Karlof, Chris and Wagner, David, "Secure Routing in Sensor Networks: Attacks and Countermeasures", Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols vol 1, issues 2-3, September 2003.

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

[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.

Appendix A. Alternatives for Delivery of Frames in a Mesh

Before settling on the mechanism finally adopted for delivery in a mesh (Section 11), several alternatives were considered. In addition to the hop-by-hop source and destination link-layer addresses, delivering a packet in a LoWPAN mesh requires the end-to-end originator and destination addresses. These could be expressed either as layer 2 or as layer 3 (i.e., IP) addresses. In the latter case, there would be no need to provide any additional header support in this document (i.e., within the LoWPAN header itself). The link-layer destination address would point to the next hop destination address while the IP header destination address would point to the final destination (IP) address (possibly multiple hops away from the source), and similarly for the source addresses. Thus, while forwarding data, the single-hop source and destination addresses would change at each hop (always pointing to the node doing the forwarding and the "best" next link-layer hop, respectively), while the source and destination IP addresses would remain unchanged. Notice that if an IP packet is fragmented, the individual fragments may arrive at any node out of order. If the initial fragment (which contains the IP header) is delayed for some reason, a node that receives a subsequent fragment would lack the required information. It would be forced to wait until it receives the IP header (within the first fragment) before being able to forward the fragment any further. This imposes some additional buffering requirements on intermediate nodes. Additionally, such a specification would only work for one type of LoWPAN payload: IPv6. In general, it would have to be adapted for any other payload, and would require that payload to provide its own end-to-end addressing information.

On the other hand, the approach finally followed (Section 11) creates a mesh at the LoWPAN layer (below layer 3). Accordingly, the link-layer originator and final destination address are included within the LoWPAN header. This enables mesh delivery for any protocol or application layered on the LoWPAN adaptation layer (Section 5). For IPv6 as supported in this document, another advantage of expressing the originator and final destinations as layer 2 addresses is that the IPv6 addresses can be compressed as per the header compression specified in Section 10. Furthermore, the number of octets needed to maintain routing tables is reduced due to the smaller size of 802.15.4 addresses (either 64 bits or 16 bits) as compared to IPv6 addresses (128 bits). A disadvantage is that applications on top of IP do not address packets to link-layer destination addresses, but to IP (layer 3) destination addresses. Thus, given an IP address, there is a need to resolve the corresponding link-layer address. Accordingly, a mesh routing specification needs to clarify the Neighbor Discovery implications, although in some special cases, it may be possible to derive a device's address at layer 2 from its

address at layer 3 (and vice versa). Such complete specification is outside the scope of this document.

Authors' Addresses

Gabriel Montenegro
Microsoft Corporation

EMail: gabriel.montenegro@microsoft.com

Nandakishore Kushalnagar
Intel Corp

EMail: nandakishore.kushalnagar@intel.com

Jonathan W. Hui
Arch Rock Corp

EMail: jhui@archrock.com

David E. Culler
Arch Rock Corp

EMail: dculler@archrock.com