Network Working Group
Internet-Draft
Expires: January 12, 2006

G. Montenegro
Microsoft Corporation
N. Kushalnagar
Intel Corp

July 11, 2005

Transmission of IPv6 Packets over IEEE 802.15.4 Networks
draft-ietf-6lowpan-format-00

Status of this Memo

Copyright Notice

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. Likewise, the provisions required for packet delivery in IEEE 802.15.4 meshes is defined. However, a full specification of mesh routing (the specific protocol used, the interactions with neighbor discovery, etc) is out of 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.

IEEE 802.15.4 defines several addressing modes. 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.

IEEE 802.15.4 allows the use of either IEEE 64 bit extended addresses or (after an association event) 16 bit addresses unique within the PAN. This document assumes use of 64 bit extended addresses, but 16 bit address support may be added in a future revision.

This document assumes that a PAN maps to a specific IPv6 link, hence it implies a unique prefix. If the PAN ID (16 bits) is included in the IEEE 802.15.4 headers, it may be possible to use it to automatically map to the corresponding IPv6 prefix. One possible method is to concatenate the 16 bits of PAN ID to a /48 in order to obtain the link prefix. Whichever method is used, the assumption in this document is that a given PAN ID maps to a unique IPv6 prefix. 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 in IEEE 802.15.4. Hence, IPv6 level multicast packets MUST be carried as link-layer broadcast frames in IEEE 802.15.4 networks. As usual, hosts learn IPv6 prefixes via router advertisements ([I-D.ietf-ipv6-2461bis]).

3. 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 fragmention and reassembly adaptation layer must be provided at the layer below IP. Such a layer is defined below in Section 4.

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 IPv6 requirements of 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 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 8.

4. Adaptation Layer and Frame Format

4.1 Link Fragmentation

All IP datagrams transported over IEEE 802.15.4 are prefixed by an encapsulation header with one of the formats illustrated below. In all cases, the encapsulation header size is 2 octets. 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. Alternatively, if the 'M' bit is on, before this actual payload, a "Final Destination" field will be present (Section 9).

If an entire IP datagram may be transmitted within a single 802.15.4 packet, it is unfragmented and the LoWPAN encapsulation SHALL conform to the format illustrated below.

NOTE: All fields marked "reserved" or "rsv" SHALL be set to zero upon transmission, and ignored upon reception.

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | LF|    prot_type        |M|rsv| Payload (or Final Destination)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Unfragmented encapsulation header format

Field definitions are as follows:

LF: This 2 bit field SHALL be zero.

prot_type: This 11 bit field SHALL indicate the nature of the datagram that follows. In particular, the prot_type for IPv6 is 1 hexadecimal. The value 2 hexadecimal is defined below for header compression (Section 8). Other protocols may use this encapsulation format, but such use is outside the scope of this document. Subsequent assignments are to be handled by IANA (Section 10).

M: This bit is used to signal whether there is a "Final Destination" field as used for ad hoc or mesh routing. If set to 1, a "Final Destination" field precedes the IPv6 packet (Section 9).

If the datagram does not fit within a single IEEE 802.15.4 frame, it SHALL be broken into link fragments. The first link fragment SHALL conform 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | LF|      prot_type      |M|datagram_tag |   datagram_size     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: First fragment encapsulation header format

The second and subsequent link fragments (up to and including the last) SHALL conform 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | LF|   datagram_offset   |M|datagram_tag |  datagram_size      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Subsequent fragment(s) encapsulation header format

Field definitions are as follows:

LF: This 2 bit field SHALL specify the relative position of the link fragment within the IP datagram, as encoded by the following table.

LF	Position	
00	Unfragmented	(Figure 1)
01	First Fragment	(Figure 2)
10	Last Fragment	(Figure 3)
11	Interior Fragment	(Figure 3)

Figure 4: Link Fragment Bit Pattern

datagram_size: This 11 bit field encodes the size of the entire IP datagram. The value of datagram_size SHALL be the same for all link fragments of an IP datagram and SHALL be 40 octets more (the size of the IPv6 header) than the value of Payload Length in the datagram's IPv6 header [RFC2460]. Typically, this field needs to encode a maximum length of 1280 (IEEE 802.15.4 link MTU as defined in this document), and as much as 1500 (the default maximum IPv6 packet size if IPv6 fragmentation is in use). Therefore, this field is 11 bits long, which works in either case.

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.

prot_type: This 11 bit field is present only in the first link fragment. For possible values, see Section 10.

M: This bit present to allow delivery of link fragment in a mesh. If set to 1, a "Final Destination" field is present as per Section 9.

fragment_offset: This field is present only in the second and subsequent link fragments and SHALL specify the offset, in octets, of the fragment from the beginning of the IP datagram. The first octet of the datagram (e.g., the start of the IP header) has an offset of zero; the implicit value of fragment_offset in the first link fragment is zero. This field is 11 bits long, as per the datagram_size explanation above.

datagram_tag: The value of datagram_tag (datagram tag) SHALL be the same for all link fragments of an IP datagram. The sender SHALL increment datagram_tag for successive, fragmented datagrams; the incremented value of datagram_tag SHALL wrap from 127 back to zero. Initial value is not defined.

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

4.2 Reassembly

The recipient of an IP datagram transmitted via more than one 802.15.4 packet SHALL use both the sender's 802.15.4 source address and datagram_tag to identify all the link fragments from a single datagram.

Upon receipt of a link fragment, the recipient may place the data payload (except the encapsulation header) within an IP datagram reassembly buffer at the location specified by fragment_offset. The size of the reassembly buffer SHALL be determined from datagram_size.

If a link fragment is received that overlaps another fragment identified by the same source address and datagram_tag, 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 fragment_offset from the encapsulation header and data_length from the 802.15.4 packet header.

Upon detection of a IEEE 802.15.4 Disassociation event, the

recipient(s) SHOULD discard all link fragments of all partially reassembled IP datagrams, and the sender(s) SHOULD discard all not yet transmitted link fragments of all partially transmitted IP datagrams.

5. Stateless Address Autoconfiguration

The Interface Identifier [RFC3513] for an IEEE 802.15.4 interface is based on the EUI-64 identifier [EUI64] assigned to the IEEE 802.15.4 device. The Interface Identifier is formed from the EUI-64 according to the "IPv6 over Ethernet" specification [RFC2464].

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 [I-D.ietf-ipv6-rfc2462bis] of an IEEE 802.15.4 interface MUST have a length of 64 bits.

6. IPv6 Link Local Address

The IPv6 link-local address [RFC3513] 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 5

7. Unicast Address Mapping

The procedure for mapping IPv6 unicast addresses into IEEE 802.15.4 link-layer addresses is described in [I-D.ietf-ipv6-2461bis].

The Source/Target Link-layer Address option has the following form when the link layer is IEEE 802.15.4.

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

Figure 6

Option fields:

Type:
1: for Source Link-layer address.
2: for Target Link-layer address.

Length: 2. This is the length of this option (including the type and length fields) in units of 8 octets.

IEEE 802.15.4 Address: The 64 bit IEEE 802.15.4 address, 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.

8. 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 that most of the time there will be only one flow, and this allows a very simple and low context flavor of header compression.

Given the very limited packet sizes, it is highly desirable to integrate layer 2 with layer 3 compression, something typically not done.

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.

Preliminary context is often required. If so, it is highly desirable to allow building it by not relying exclusively on the in-line negotiation phase. For example, if we assume there is some manual configuration phase that precedes deployment (perhaps with human involvement), then one should be able to leverage this phase to set up context such that the first packet sent will already be compressed.

Any new packets formats required by header compression reuse the basic packet formats defined in Section 4 by using different values for the prot_type (defined below).

8.1 Encoding of IPv6 Header Fields

However, it is possible to use header compression even in advance of setting up the customary state. Thus, the following common IPv6 header values may be compressed from the onset: Version is IPv6, both IPv6 source and destination are link local, the IPv6 bottom 64 bits can be inferred from the layer two source and destination, the packet length can be inferred from the layer two, both the Traffic Class and the Flow Label are zero, and the Next Header is UDP, ICMP or TCP. Thus, the IPv6 header info that always needs to be carried 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 8.3.1). Thus this common IPv6 header 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 (assigned a prot_type value of 2 hexadecimal). It uses the "HC1 encoding" field (8 bits) to encode the different combinations as 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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | HC1 encoding  |     Non-Compressed fields follow...           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: 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 8.3.

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

PI: Prefix carried in-line (Section 8.3.1).
PC: Prefix compressed (link-local prefix assumed).
II: Interface identifier carried in-line (Section 8.3.1).
IC: Interface identifier elided (to be derived from the corresponding link-layer address). If applied to the destination interface identifier when routing in a mesh (Section 9), the corresponding link-layer address is that found in the "Final Destination" field (Figure 9).

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).

8.2 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 8) 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 8.3.2).

```
                     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 8: 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 8.3.2)
- 1: Compressed to 4 bits. The actual 16-bit source port is obtained by calculating: P + short_port value. P is a predetermined port number with value TBD. The short_port is expressed as a 4-bit value which is carried "in-line" (Section 8.3.2)

UDP destination port (bit 1):
- 0: Not compressed, carried "in-line" (Section 8.3.2)
- 1: Compressed to 4 bits. The actual 16-bit destination port is obtained by calculating: P + short_port value. P is a predetermined port number with value TBD. The short_port is expressed as a 4-bit value which is carried "in-line" (Section 8.3.2)

Length (bit 2):
- 0: not compressed, carried "in-line" (Section 8.3.2)
- 1: compressed, length computed from IPv6 header length information (similar to how the length of the header is calculated in TCP

Reserved (bit 3 through 7)

Note: TCP, ICMP HC2 formats TBD.

8.3 Non-Compressed Fields

8.3.1 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.

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 7), 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 8.1): 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.

8.3.2 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.

9. Packet Delivery in a Link-Layer Mesh

IEEE 802.15.4-2003 [ieee802.15.4] does not define a mesh routing capability. Nevertheless, it is expected that most 802.15.4 networks will use mesh routing. In such cases, an ad hoc or mesh routing procotol populates the devices' routing tables. A device that wishes to send a packet may, in such cases, use other intermediate devices as forwarders towards the final destination. In order to achieve such packet delivery using unicast, it is necessary to include the final destination in addition to the hop-by-hop destination. This final destination may be expressed either as a layer 2 or as an IP (layer 3) address.

In the latter case, there is no need to provide any additional header support in this document (i.e., at the sub-IP layer). The link-layer destination address points to the next hop destination address while the IP destination address points to the final destination (IP) address (that may be multiple hops away from the source). Thus, while forwarding data, the single-hop destination address changes hop-by-hop pointing to the "best" next hop, while the destination IP address remains unchanged.

If creating a mesh at the link-layer (layer 2), there is a need to include the link-layer final destination address within the packet. The advantage of expressing the final destination as a layer 2 addresses is that the IPv6 destination address can be compressed as per the header compression specified in Section 8, thus saving 8 octets. Another advantage is that the the number of octets needed to maintain routing tables is reduced. A disadvantage is that applications do not address packets to link-layer destination addresses, but to IP (layer 3) addresses. Thus, given an IP address, there is a need to resolve the corresponding link-layer address. A mesh routing specification needs to clarify the Neighbor Discovery implications, although in some special cases, it may be possible to derive one address from the other. Such complete specification is outside the scope of this document.

This document merely defines how to effect packet delivery in a mesh, given a target link-layer address.

This is the purpose of the 'M' bit that immediately follows the 'prot_type' field. If the 'M' bit is set, there is a "Final Destination" field included in the packet immediately following the current header (e.g., possibly preceding any existing header compression fields). This implies that the "Final Destination" field will immediately follow an unfragmented packet as per Figure 1 (i.e., preceding the IPv6 Header), or immediately following the first fragment header as per Figure 2.

If a node wishes to use a forwarder to deliver a packet, it puts the forwarder's link-layer address in the link-layer destination address field. It must also set the 'M' bit, and include a "Final Destination" field with the final destination's link-layer address. Similarly, if a node receives a frame with the 'M' bit set, it must look at the "Final Destination" field to determine the real destination. Upon consulting its routing table, it determines what the next hop towards that destination should be. The node then reduces the "Hops Left" field. If the result is zero, the node discards the packet. Otherwise, it puts the next hop's address in the link-layer destination address field, and transmits the packet. If upon examining the "Final Destination" field the node determines that it has direct reachability, it removes the "Final Destination" field, sets that final address as the link-layer destination address, and transmits the packet.

The "Final Destination" field is defined as follows:

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |S| Hops Left   |      Address of final destination             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Final Destination Field

Field definitions are as follows:

S: This bit field SHALL be zero. Future revisions will use this bit to signal the use of a short 16 bit address instead of the default IEEE extended 64 bit address format.

Hops Left: This 7 bit field SHALL be decremented by each forwarding node before sending this packet towards its next hop. The packet is discarded if Hops Left is decremented to 0.

Address: This is the final destination's link-layer address. This document assumes that this field is 64 bits long, but a future revision may add support for short addresses (16 bits).

10. IANA Considerations

This document creates a new IANA registry for the prot_type (Protocol Type) field shown in the packet formats in Section 4. This document defines the values 1 and 2 hexadecimal for IPv6 and the LOWPAN_HC1 header compression format, respectively. Future assignments in this field 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 defines this field to be 5 bits long. The value 0 being reserved and not used, this allows for a total of 31 different values. If there is a need for more assignments, future specifications may lengthen this field, e.g., by overloading the packet format in Figure 2 (Section 4).

11. Security Considerations

The method of derivation of Interface Identifiers from 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 referred to 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 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 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).

12. 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 and Samita Chakrabarti.

13. Changes

Changes from version draft-montenegro-lowpan-ipv6-over-802.15.4-02.txt to version draft-ietf-6lowpan-format-00.txt are as follows:

The LoWPAN encapsulation was modified to allow 11 bits of protocol type (prot_type field). Because of this, the minimum overhead grew from 1 octet to 2 octets. This was done in order to allow more protocol types as the previous format started with a field only 5 bits wide. Whereas growing it to 7 bits was possible in the future, this would always entail 2 octets of overhead for the longer protocol types to be used.

The 'M' bit had been left out of the 3rd packet format (for subsequent fragments). Corrected this oversight. This means that the fragment tag lost one bit.

Sundry editorial changes.

14. References

14.1 Normative References

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

[I-D.ietf-ipv6-2461bis]
Narten, T., "Neighbor Discovery for IP version 6 (IPv6)",

draft-ietf-ipv6-2461bis-03 (work in progress), May 2005.

[I-D.ietf-ipv6-rfc2462bis]
Thomson, S., "IPv6 Stateless Address Autoconfiguration", draft-ietf-ipv6-rfc2462bis-08 (work in progress), May 2005.

[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.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

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

14.2 Informative References

[I-D.ietf-ipngwg-icmp-v3]
Conta, A., "Internet Control Message Protocol (ICMPv6)for the Internet Protocol Version 6 (IPv6) Specification", draft-ietf-ipngwg-icmp-v3-06 (work in progress), November 2004.

[I-D.ietf-ipv6-node-requirements]
Loughney, J., "IPv6 Node Requirements", draft-ietf-ipv6-node-requirements-11 (work in progress), August 2004.

[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.

[RFC1042] Postel, J. and J. Reynolds, "Standard for the transmission of IP datagrams over IEEE 802 networks", STD 43, RFC 1042, February 1988.

[RFC3439] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, December 2002.

[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.

Authors' Addresses

Gabriel Montenegro
Microsoft Corporation

Email: gabriel_montenegro_2000@yahoo.com

Nandakishore Kushalnagar
Intel Corp

Email: nandakishore.kushalnagar@intel.com

Intellectual Property Statement

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.