Network Working Group   G. Montenegro
Internet-Draft   Sun Microsystems, Inc.
Expires: June 30, 2005   December 30, 2004

Transmission of IPv6 Packets over IEEE 802.15.4 Networks
draft-montenegro-lowpan-ipv6-over-802.15.4-00.txt

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.

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.

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].

2. Maximum Transmission Unit

The MTU size for IPv6 packets over IEEE 802.15.4 is 1280 octets. However, a full 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 3.

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 7.

NOTE: In traditional IEEE 802 applications, a further 8 octets are taken up by LLC/SNAP encapsulation [RFC1042], which would leave only 73 octets for upper layer protocols (e.g., IP). SNAP encapsulation is not used in this specification. Any heartburn about this? Must think about compatibility with other applications (what do these do?). To guarantee interoperability, we might want to add the SNAP header. It's just more fixed overhead, as instead of following with the ether_type for IPv6 (and overloading the version field as per the hack in RFCs 1144 and 2507), we would want to follow the SNAP header with a new identifier for the adaptation layer defined below.

3. Adaptation Layer and Frame Format

3.1 Link Fragmentation

All IP datagrams transported over IEEE 802.15.4 are prefixed by an encapsulation header with one of the formats illustrated below.

If an entire IP datagram may be transmitted within a single 802.15.4 packet, it is unfragmented and the first octet of the data payload SHALL conform to the format illustrated below. In this case, the overhead is 1 octet. It is expected that this will be, by far, the most common case.

NOTE: All fields marked "reserved" or "rsv" SHALL be zero.

```
                     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 |    IPv6 packet begins...                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Unfragmented encapsulation header format

Field definitions are as follows:

LF: This 2 bit field SHALL be zero.

prot_type: This 6 bit field SHALL indicate the nature of the datagram that follows. In particular, the prot_type for IPv6 is 1 hexadecimal. 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 8).

NOTE: This field serves a purpose similar to that of the PPP DLL or ethertype protocol numbers (16 bits). However, in the interest of reducing the overhead in the common case, here we only have 6 bits. Assuming that we do not use the value zero, this leaves 63 type assignments in total. It is apparent that this may be enough. But in case it is not, it is important to know that it is possible to grow beyond these 6 bits. One way to do so is to assume that the actual field holds 8 bits, which leaves plenty of possibilities for future assignments. In such a case, the above format could only be used with the first 63 types assignments. Use of types beyond the initial 63 assignments would require use of the frame format below. This format, defined below to transmit the *first* fragment, can be overloaded to mean "first *and* last" (i.e., unfragmented). This can be accomplished by using a frag_label of zero (otherwise illegal), and/or simply in an implicit fashion via the datagram_size information. Accordingly, it seems prudent to leave a "rsv" field in front of the prot_type field in the frame below, pending further discussion.

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|rsv  |  prot_type    |  frag_label   |   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    |  frag_label    |  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
0	Unfragmented
1	First
2	Last
3	Interior

Figure 4: Link Fragment Bit Pattern

datagram_size: The encoded 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 field is present only in the first link fragment and SHALL have a value of 1 hexadecimal which indicates an IPv6 datagram. See Section 8.

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 (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_label: The value of datagram_label (datagram label) SHALL be the same for all link fragments of an IP datagram. The sender SHALL increment datagram_label for successive, fragmented datagrams; the incremented value of datagram_label SHALL wrap from 255 back to one. The value zero is not used.

NOTE: The value zero is reserved as per the note under Figure 1. This may allow for a future overloading of the "first fragment" header to also mean "first and last fragment", thus allowing the use of extended protocol type numbers (8 bits instead of 6 bits).

All IP datagrams SHALL be preceded by one of the encapsulation headers described above. This permits uniform software treatment of datagrams without regard to the mode of their transmission.

3.2 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 frag_label 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 may be determined from datagram_size.

If a link fragment is received that overlaps another fragment identified by the same source address and frag_label, 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.

4. 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.

5. 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

6. 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 (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.

7. Header Compression

The header compression for IPv6 packets over IEEE 802.15.4 is as follows:

TBD

8. IANA Considerations

This document creates a new IANA registry for the prot_type (Protocol Type) field shown in the packet formats in Section 3. This document

defines the value 1 hexadecimal for IPv6. 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 6 bits long. The value 0 being reserved and not used, this allows for 63 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 3).

9. 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.

10. Acknowledgements

Thanks to the authors of RFC 2464 and RFC 2734, as parts of this document are patterned after theirs. Thanks also to Geoff Mulligan and Nandakishore Kushalnagar for discussions which have helped shaped this document.

11. References

11.1 Normative References

[I-D.ietf-ipv6-2461bis]
Narten, T., "Neighbor Discovery for IP version 6 (IPv6)", draft-ietf-ipv6-2461bis-01 (work in progress), October 2004.

[I-D.ietf-ipv6-rfc2462bis]
Thomson, S., "IPv6 Stateless Address Autoconfiguration", draft-ietf-ipv6-rfc2462bis-07 (work in progress), December 2004.

[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.

11.2 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.

[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.

Author's Address

Gabriel Montenegro
Sun Microsystems, Inc.

Phone:
EMail: gab@sun.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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