

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 28, 2011

A. Ayadi
D. Ros
L. Toutain
Telecom Bretagne
October 25, 2010

TCP header compression for 6LoWPAN
draft-aayadi-6lowpan-tcphc-01

Abstract

This document describes LOWPAN_TCPHC, a scheme for compressing the header of Transmission Control Protocol (TCP) segments, in order to reduce the overhead on low-power and lossy networks. It also specifies the LOWPAN_TCPHC header fields for the transmission of TCP segments over IPv6 for Low-power Wireless Personal Area Networks (6LoWPAN). In many cases, the 20 bytes of the mandatory TCP header can be compressed into as little as 6 bytes.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 28, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Related Work	4
1.2. Terminology	5
2. Protocol Overview	6
2.1. Connection Initiation	8
2.2. The LOWPAN_TCPHC context	10
2.2.1. LOWPAN_TCPHC context structure	10
2.2.2. Context management	12
2.3. Loss detection and retransmissions	12
3. Transmission Control Protocol	13
3.1. TCP headers fields	14
3.2. TCP Header Options	15
4. TCP header fields compression	15
4.1. TCP ports	16
4.2. Flags	16
4.3. Sequence and Acknowledgment numbers	16
4.4. Window	17
4.5. Urgent Pointer	17
5. LOWPAN_TCPHC Packet Format	17
5.1. TCP segments types	17
5.2. LOWPAN_TCPHC Format	18
5.3. Examples of compressed TCP headers	19
5.3.1. Compressed header	19
6. TCP Option Compression	19
6.1. Selective Acknowledgment option	20
6.2. Timestamp option	20
7. Acknowledgments	21
8. References	21
Authors' Addresses	23

1. Introduction

The 6LoWPAN Working Group [RFC4919] has proposed LOWPAN_IPHC [I-D.ietf-6lowpan-hc], a new version of the LOWPAN_HC1 header compression mechanism [RFC4944] which reduces the IPv6 header to about 3-5 bytes. In [I-D.ietf-6lowpan-hc], a header-compression method for the transport layer (LOWPAN_NHC) has also been introduced, however, only a UDP [RFC0768] datagram compression mechanism is specified.

UDP header compression is useful for 6LoWPAN because many Low-power and Lossy Network (LLN) applications are fault-tolerant and do not require 100% reliability. However, other applications and services such as SSH and HTTP require a reliable service from the transport layer that cannot be offered by UDP. Moreover, some usage scenarios of LLNs, such as health, military and security applications, impose strong reliability constraints. Also, in some usage cases (e.g., sending a software update to a wireless node, or sending a query requesting a specific information from the wireless node) there is a need for a reliable data transport.

In this document, we focus on the Transmission Control Protocol (TCP) [RFC0793], which carries most of the traffic in IP networks. TCP is a connection-oriented, end-to-end reliable transport protocol. To ensure end-to-end reliability, TCP must recover from packet corruption, loss or out-of-order delivery. This is achieved by assigning a sequence number to each transmitted byte (done by the TCP source), by requiring a positive acknowledgment (ACK) from the TCP destination, and by retransmitting lost or corrupted packets.

In the context of 6LoWPAN networks, the size of TCP headers may induce a large overhead, especially for link-layer technologies that use small frames. For instance, a pure TCP acknowledgment (i.e., a TCP ACK carrying no data) without any TCP options represents 25% of the payload of an IEEE 802.15.4 typically [IEEE 802.15.4] MAC frame. In addition, TCP pure ACKs represent roughly 33% of the total number of segments exchanged in a TCP session (this figure may go up to roughly 50% if the Delayed ACK mechanism [RFC1122] is not used). This suggests that the use of header-compression mechanisms for TCP may result in important performance gains, in terms of used bandwidth and/or energy spent for frame transmission.

A TCP header compression algorithm for 6LoWPAN should respect some requirements: (a) Efficiency (the scheme must provide low overhead in all cases), (b) Transparency (the resulting header after a compression and decompression should be identical to the original header), (c) Reordering tolerance (the scheme should be able to decompress compressed segments correctly even when segments arrive

with a moderate reordering).

The goal of this document is thus to define a TCP header compression scheme for 6LoWPAN, called LOWPAN_TCPHC, which allows to significantly reduce the TCP overhead. The TCP header compression and decompression is performed in the edge router between the 6LoWPAN and the external IP network. The compression scheme can also be used between two 6LoWPAN nodes for machine-to-machine communications. This document defines also an encoding format for LOWPAN_TCPHC header compression. Such mechanism and packet format aims at making TCP a more viable proposition for 6LoWPAN networks. Moreover, the LOWPAN_TCPHC mechanism can be used with LOWPAN_IPHC [I-D.ietf-6lowpan-hc] and thus reduce all header overheads to about seven to ten bytes instead of 60 bytes.

The proposed scheme does not compress TCP control messages at the connection establishment phase. Those TCP segments are used to exchange a context identifier. Such context identifier replaces the port numbers in subsequent TCP segments, as a means of identifying a given TCP session. Some TCP options, such as Timestamp [RFC2018] and SACK [RFC1323] [RFC2883], are supported (i.e., compressed) by the mechanism, while other options, unlikely to be required/used in 6LoWPANs, are omitted.

1.1. Related Work

This section presents prior work on TCP/IP header compression. In particular, we will briefly describe three existing TCP header compression algorithms. A more detailed discussion of these algorithms can be found in [RFC4996].

One of the first TCP/IP header compression methods was Compressed TCP (CTCP), proposed by Jacobson [RFC1144]. Jacobson's header compression algorithm distinguishes between dynamic fields and static fields. The static fields (e.g., source address, source port, ...) are sent in two situations: when initiating a connection, and when refreshing the context after a loss of synchronization. CTCP proposes to send the difference between the current and the previous value of dynamic fields (e.g., sequence number, acknowledgment number). When the synchronization is lost between the compressor and the decompressor, the TCP sender sends a segment with a regular header to refresh the context. Experimental studies [Perkins et al.] [Srivastava et al.] [Wang04] have shown that the performance of Jacobson's algorithm may degrade significantly in noisy/lossy network environments. An important disadvantage of CTCP is that it does not support TCP options, some of which are ubiquitous nowadays [Medina05].

IPHC [RFC2507] enhances Jacobson's TCP header compression by introducing a mechanism, called TWICE, to repair incorrectly-decompressed headers. TWICE is most efficient when applied to data-carrying TCP segments. [RFC2507] also describes a mechanism for explicitly requesting the transmission of less-compressed or uncompressed headers. such mechanism is especially suited for pure TCP acknowledgments. Note however that IPHC does not actually provide a compression method for TCP options; changing option fields are carried in compressed headers, but without any compression. Also, the header request mechanism may be unsuited for lossy 6LoWPAN networks, which low bit rates and strong energy constraints are at odds with any additional signaling overhead. LoWPAN_TCPHC enhances IPHC by defining an adapted header compression of TCP for LLNs by sending least significant bytes instead of sending a delta value. Moreover, LoWPAN_TCPHC completes IPHC by defining header compression schemes for the mostly used TCP options.

ROHC-TCP [RFC4996] improves on [RFC2507] by providing a new method for compressing all TCP header fields, including the TCP options. ROHC-TCP proposes also to start compressing packets starting from the SYN segments, using parameters from previous or simultaneous connections. This may offer noticeable improvements in performance when most TCP flows are short-lived, i.e., composed of a small number of data segments. Nevertheless, the ROHC-TCP algorithm is fairly complex and its memory requirements may not be met by small, constrained devices.

The algorithm described in this document, LOWPAN_TCPHC, supports features like the compression of TCP options and FIN segments, and at the same time it is relatively simple and easy to implement in memory- and CPU-constrained devices.

1.2. Terminology

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

This section defines general terms related to TCP/IPv6 header compression [RFC2507] [RFC4995] and to the 6LoWPAN architecture [RFC4919] used in this specification.

- o Subheader: An IPv6 header, a UDP header, or a TCP header.
- o Header: A chain of subheaders.
- o Compression: The act of reducing the size of a header by either removing or reducing the size of header fields. This is done in a way such that a decompressor can reconstruct the header if its context state is identical to the context state used when

compressing the header.

- o Decompression: The act of reconstructing a compressed header.
- o Static fields: These fields are expected to be constant throughout the lifetime of the TCP connection. Static information must be communicated once in some way.
- o Dynamic fields: These fields are expected to vary between different TCP segments, either randomly within a limited range or in some other manner.
- o Context identifier (CID): A small unique number identifying the context that should be used to decompress a compressed header. Carried in full headers and compressed headers.
- o Context: The state used by the compressor to compress a header, and by the decompressor to decompress a header. The context is given by the uncompressed version of the last header sent (compressor) or received (decompressor) over the link, except for fields in the header that are included "as-is" in compressed headers, or that can be inferred from e.g. the size of the link-layer frame.
- o Full header: An uncompressed header that updates or refreshes the context for a packet stream. It carries a CID that will be used to identify the context.
- o Regular header: A normal, uncompressed header. It does not carry any CID.
- o Compressed header: A header in which all the static fields are elided, and all the dynamic fields are sent compressed.
- o Incorrect decompression: When a decompressed header does not match the corresponding original, uncompressed header. Usually due to mismatching contexts between the compressor and decompressor, caused by e.g. bit errors during the transmission of the compressed header, or by packet loss.
- o IEEE 802.15.4: A low-power, low-bandwidth link layer protocol.
- o LoWPAN host: A node that only sources or sinks IPv6 datagrams. Referred to as a host in this document.
- o LoWPAN router: A node that forwards datagrams between arbitrary source-destination pairs using a single 6LoWPAN interface, performing IP routing on that interface.
- o LoWPAN edge router (ER): An IPv6 router that interconnects the 6LoWPAN to another IP network. Referred to as an Edge Router in this document.
- o LoWPAN node: A node that composes a 6LoWPAN, referring to both hosts and routers. Simply called a Node in this document.

2. Protocol Overview

This section gives an overview of the TCP header compression mechanism for 6LoWPAN (LOWPAN_TCPHC). The main purpose of LOWPAN_TCPHC is to reduce the protocol header overhead, with the

intent of reducing both bandwidth usage and energy consumption due to packet transmissions.

Indeed, the LOWPAN_TCPHC allows establishing TCP connections between an external IP host and a LoWPAN host, and also between two LoWPAN hosts. This former type of connection is performed by an Edge Router (ER) which links the 6LoWPAN to an external IPv6-based network. Figure 1 shows a typical 6LoWPAN topology with which two edge router create a bridge between the LoWPAN network and the external IP network. The path between a LoWPAN node to the external network may change following the movement of the node or the update of routing tables.

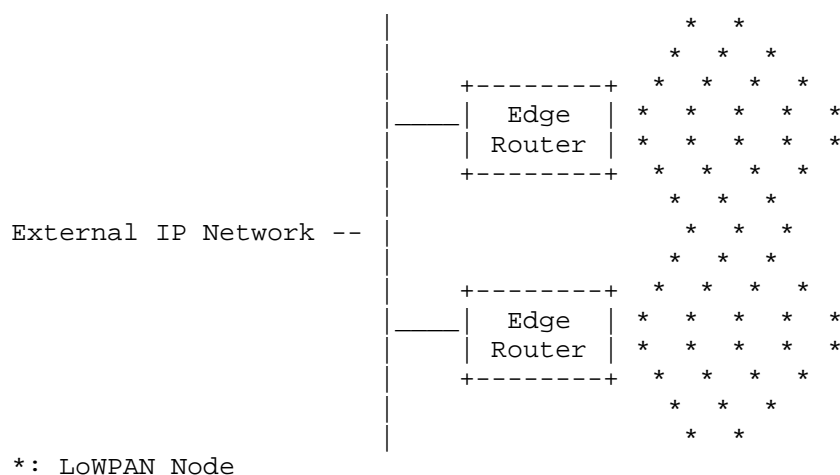


Figure 1: A 6LoWPAN network

The compression and decompression mechanisms are implemented on the edge routers and on the LoWPAN hosts. The ERs create and maintain the contexts of all TCP connections. Figure 2 and Figure 3 show sequence diagrams of a connection establishment between a LoWPAN host and an external IP host. The external IP host sends and receives regular TCP segments, whereas the LoWPAN host sends and receives segments with compressed headers or full headers.

The LOWPAN_TCPHC algorithm does not compress TCP control messages at the connection establishment phase. These segments are used to exchange the context identifier (CID) and allow the ER to create a context using the TCP header fields. The LOWPAN_TCPHC algorithm supports useful TCP options for LLNs. Supported TCP options that are negotiated in the two first messages are sent in a full header format. Whereas, the remaining supported options are sent compressed within the TCP segments. The LOWPAN_TCPHC algorithm defines a

compression mechanism for TCP SACK and Timestamp options.

The TCP connection may also be established between two LoWPAN hosts for Machine-to-Machine communications. In this case, the context should be shared between the two LoWPAN hosts. If a packet is dropped due to loss, then the mechanism refreshes the context by retransmitting lost segments using the mostly compressed header format.

The compression protocol uses two different header formats. For the TCP opening phase and for error management, the TCP segments must be sent with a full header. These segments contain the Context Identifier (CID) which will be used to identify the connection during the transfer phase. The CID replaces the two port numbers, hence avoiding to send the port numbers in every packet. The CID value and its size are set by the LoWPAN host if the TCP connection is between a LoWPAN host and an external IP host. Otherwise (i.e., a TCP connection between two LoWPAN hosts), the CID value and its size are set by the LoWPAN host that has opened the connection. For the TCP connection between a LoWPAN host and an external IP host, the couple (CID, Ipv6 address) must be unique.

The second kind of header is the compressed header in which all static fields are elided, and all dynamic fields are compressed. Depending on how they change with respect to the last sent segment, dynamic fields may be compressed fully (i.e., elided) or partially (i.e., only a portion of a field is sent, containing the bytes that have changed).

The last kind of packet is sent when a packet is lost, corruption or reordering is detected by the TCP receiver. This segment is called mostly compressed header and contains dynamic fields with all bytes uncompressed, while its static fields are elided. This kind of segment should be sent after a loss of synchronization between the compressor and the decompressor.

2.1. Connection Initiation

This section gives an overview of TCP connection initiation with LOWPAN_TCPHC.

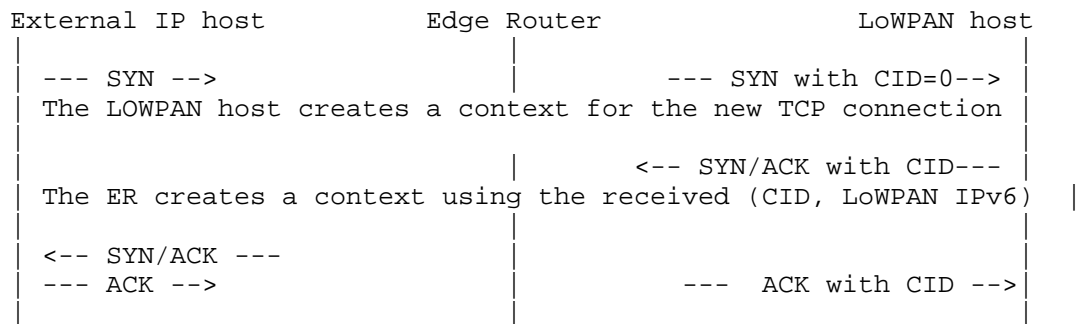


Figure 2: TCP Connection initiated by an external IP-host

Figure 2 shows an example of a connection initiation scenario started by an external IP host. In this scenario, an external host sends a SYN segment trying to establish a connection with a LoWPAN host. The SYN message can include options, some of which may be eliminated by the Edge Router (i.e., those options not supported by the LOWPAN_TCPHC mechanism). The ER sends the SYN segment in a full header message with a CID equal to zero. Upon the reception of the SYN segment, the LoWPAN node sends a SYN/ACK segment including a new CID, set to the smallest available CID value. The ER creates a new context with the received information from the SYN/ACK segment.

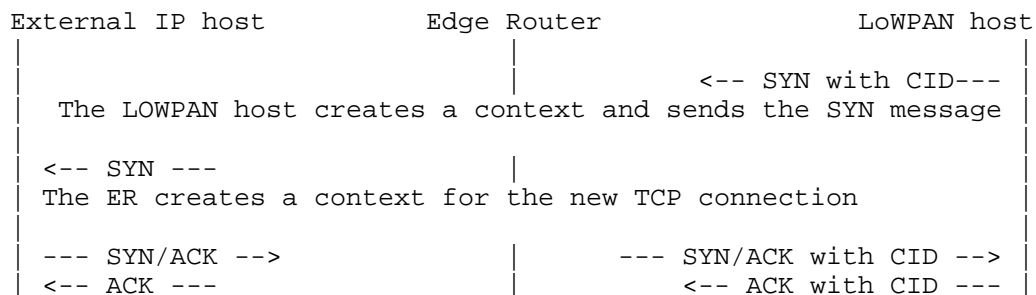


Figure 3: TCP Connection initiated by a LoWPAN host

Figure 3 gives an example of a TCP connection initiated by a LoWPAN host. The first SYN segment is sent with the full header with a CID value equal to smallest available value of CIDs. Contrary to the first case, the ER creates a context upon the reception of the SYN message from the LoWPAN host.

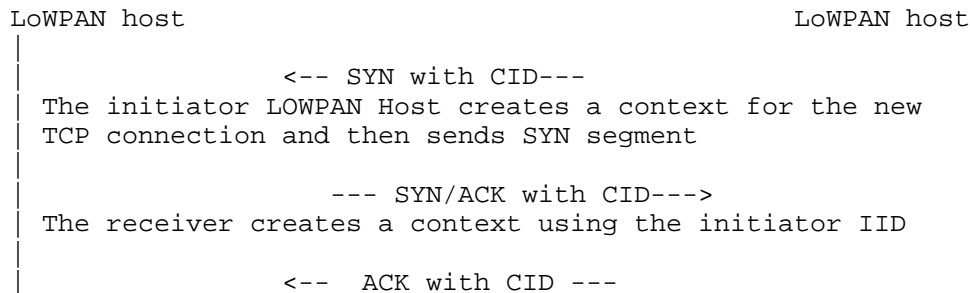


Figure 4: TCP/IPv6 Connection between two LoWPAN hosts

Figure 4 presents a TCP connection initiated by a LoWPAN node to another LoWPAN node. The LoWPAN host sends in the TCP SYN segment with a CID equals to the smallest available CID value.

The CID value is chosen by the first LoWPAN host that is involved in establishing the TCP connection. This way, it is easy to ensure the unicity of the CID. At the same time, this simplifies the context management at the ERs. Moreover, if all CID values are used, the initiator should increase the CID field length from 8 bits to 16 bits.

2.2. The LoWPAN_TCPHC context

2.2.1. LoWPAN_TCPHC context structure

The context includes some TCP header fields that are needed by the compressor and the decompressor algorithm. Figure 5 lists the contents of a LoWPAN_TCPHC context.

Field	Description	Length
CID	Context Identifier	16 bits
Address	LoWPAN IPv6 address	128 bits
SrcPort	TCP Source Port Number	16 bits
DstPort	TCP Destination Port Number	16 bits
seq_rcv	Sequence number in incoming segments	32 bits
ack_rcv	Acknowledgment number in reception	32 bits
wnd_rcv	Window size in reception	16 bits
seq_snd	Sequence number in reception	32 bits
ack_snd	Acknowledgment number in reception	32 bits
wnd_snd	Window size in reception	16 bits
State	Context state	4 bits

Figure 5: Structure of a LOWPAN_TCPHC context

The address field saves the Ipv6 address of the 6LoWPAN node. If the TCP connection is established between two 6LoWPAN nodes, the Ipv6 address of the initiator is saved in the context.

State field indicates in which state a TCP connection is. This field is especially needed by the ERs. The possible states of a context are: closed, using, closing, fin_1, fin_2, fin_3, and shutting.

The CID and Ipv6 address are utilized to identify the connection for compressed headers. The SrcPort and DstPort, together with the Ipv6 address, are used to identify the connection for full headers.

The seq_rcv, ack_rcv, and wnd_rcv are used to store the dynamic fields of the last incoming segment (except retransmitted segment). While the seq_snd, ack_snd, and wnd_snd are the dynamic fields of the last outgoing segment (except retransmitted segment).

2.2.2. Context management

This section describes the context management in 6LoWPAN when LOWPAN_TCPHC is used. The edge router to which a LoWPAN node host is attached may change over time, due to route instability or to host mobility. However, this change should not break the TCP communication. To ensure the TCP communication despite the change of ER, the ERs should share the contexts of current connections. So, even if a 6LoWPAN node changes its attached ER, the new ER should continue to compress the segments using the same context. Context exchange and management between ERs is out of the scope of this document.

The edge router should free a context when a TCP connection is finished (e.g., reception of FIN control messages). The Edge Router can also free a connection after a silent period (i.e., when no messages are exchanged after a certain period of time).

The ER may remove the context of a TCP connection that is not yet closed. In this case, after receiving a new data segment, the ER SHOULD reply by sending a RST segment to the sender.

2.3. Loss detection and retransmissions

In this section, we present how the LOWPAN_TCPHC mechanism should react when a segment is lost or is assumed to be lost. The loss is handled when the TCP ACK segment is not received within the RTO. The ER handles a retransmission by scanning the sequence numbers. The ER should send a mostly compressed header segment when it receives an already sent segment. This mechanism allows updating the context on both sides after a packet loss. We assume that the 6LoWPAN has a low bit rate, and also that nodes are memory-constrained and thus the TCP window size is probably limited to a few segments. In this case, the loss of synchronization will likely not lead to a burst of losses. For this reason, this document does not present a refresh algorithm to update the context between the compressor and the decompressor.

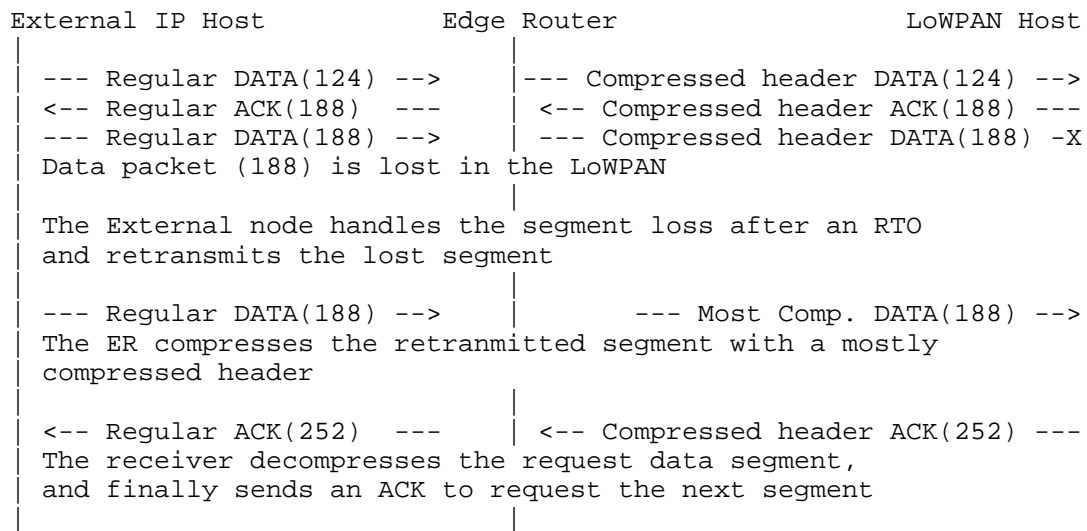


Figure 6: Loss detection in 6LoWPAN

Figure 6 shows a scenario where a TCP segment is lost in the 6LoWPAN. After an RTO the external IP host retransmits the lost segment. Upon the reception of the retransmitted segment, the ER produces a mostly compressed header allowing the LoWPAN node to decompress the segment.

3. Transmission Control Protocol

This section presents more details on TCP and its header fields. As it has been defined in [RFC0793], The TCP is a connection-oriented, end-to-end reliable transport protocol mostly used in IP-based networks. The TCP is able to transfer a continuous stream of bytes in each direction between two end-points by packing some number of bytes into segments for transmission through IP-based network.

To ensure the end-to-end reliability, the TCP must recover data that is damaged, lost or delivered out-of-order. This is achieved by assigning a sequence number to each transmitted byte (done by the TCP source), and by requiring a positive acknowledgment (ACK) from the TCP destination. If the ACK is not received within the timeout interval, the data segment is assumed to be lost. Then, the source TCP should retransmit it. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and eliminate duplicates. Damaged segments are handled by adding a checksum to each segment transmitted, checking it at the receiver and rejecting damaged segments.

3.1. TCP headers fields

In this section, we present a short description of TCP header fields and how they are handled by the compression mechanism. [RFC4413] provides a detailed description of TCP header and TCP header options.

- o Source port (16 bits): This field identifies the sending port. This field will be replaced by the CID in compressed headers.
- o Destination port (16 bits): This field identifies the receiving port. This field will be replaced by the CID in compressed headers.
- o Sequence Number (32 bits): The sequence number of the first data byte in this segment (except when SYN flag is set). If SYN is present the sequence number is the initial sequence number (ISN) and the first data byte is ISN+1. Only the bytes that change (respect to the previous segment) will be sent. If this field does not change, nothing should be sent.
- o Acknowledgment number (32 bits): If the ACK flag is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent. Only changed bytes of this field according to the last sent segment will be sent. If this field does not change, nothing should be sent.
- o Reserved (4 bits): Reserved for future use. Must be zero. This field should not be sent in compressed segments.
- o Flags (8 bits):
 - URG (1 bit): Urgent Pointer field contains a valid value.
 - ACK (1 bit): Acknowledgment field contains a valid value.
 - PSH (1 bit): Push.
 - RST (1 bit): Reset the connection.
 - SYN (1 bit): Synchronize sequence numbers.
 - FIN (1 bit): No more data from sender.
 - CWR (1 bit): Congestion window reduced.
 - ECE (1 bit): Echo the 'congestion experienced' signal in the IP header.
- o Window (16 bits): The number of data bytes, beginning with the one indicated in the acknowledgment field, the sender of this segment is willing to accept. This field is compressed and only the bytes that have changed are sent.
- o Checksum (16 bits) The 16-bit checksum field is used for error-checking of the header and data. This field is not compressed by LOWPAN_TCPHC.
- o Urgent Pointer (16 bits): This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the byte following the urgent data. This field is only interpreted in segments with the URG flag set to 1. This field is rarely used in TCP communications. For this reason, the URG flag

and Urgent Pointer are sent if and only if they are set. In this case, the segment is sent with a full header.

3.2. TCP Header Options

Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. Options are considered for the checksum calculation. The most used TCP options are indicated below:

- o End of Option List (8 bits): indicates the end of the option list.
- o No-Operation (8 bits): may be used between options.
- o Maximum Segment Size (32 bits): the maximum segment size at the TCP which sends this segment.
- o Window Scale Option (24 bits): indicates that the sending TCP end-host is prepared to perform both send and receive window scaling.
- o SACK-Permitted (16 bits): ask for using SACK option.
- o SACK (80-320 bits): selective acknowledgment.
- o Timestamp (80 bits): used to compute RTT.

The TCP header padding is used to ensure that the TCP header ends and data begins on the 32-bit boundary.

4. TCP header fields compression

In this section, we define the LOWPAN_TCPHC specifications for TCP header compression for LLNs. LOWPAN_TCPHC initiates the compression algorithm by exchanging a context identifier at the beginning of the connection. The compressor and decompressor store most fields of the first full headers as a context.

As described in [RFC2507], the context consists of the header fields whose values are constant or regularly increasing. The dynamic fields should be elided because they are the same with respect to the previous header. In fact, it is more efficient to send fewer bits, which are the difference from previous value comparing to the sending of the absolute value. The LOWPAN_TCPHC mechanism is based on sending only those fields or parts of fields that do change with respect to the previously sent packet. That is why, the TCP receiver should save the last received segment fields to decompress the next one. For example, the two first bytes of the sequence number field can be elided if they are equal to the value of the previous segment.

If a TCP segment is lost and must be retransmitted, the retransmitted segment should be sent with a mostly compressed header. Then, the TCPHC receiver updates the context.

The TCP compression format is shown in Figure 10. The three first bits are used to dispatch between different types of segments.

LOWPAN_TCPHC uses two bytes which contain the uncompressed flags of TCP.

Source and destination port numbers are omitted and replaced by a context identifier. The latter should be sent in connection initiation messages and replaces the source and destination port numbers.

The checksum is not compressed and is used by the receiver to check if the decompressed TCP segment is received correctly. To reduce the TCP header length, only these fields which have been changed between two successive segments need to be sent to the receiver. Thus, based on the previously received segment, the receiver reconstructs the original TCP header.

The urgent pointer field is transmitted only when the urgent bit is set. When a receiver sends a duplicated acknowledgment, LOWPAN_TCPHC can compress the TCP header down to six bytes (2 bytes LOWPAN_TCPHC, 1 byte CID, 1 byte acknowledgment number, 2 bytes Checksum).

4.1. TCP ports

These fields are part of the definition of a stream and they must be constant for all packets in the stream. TCP port numbers can be elided in TCP compressed segments and replaced by a context identifier (CID). The context identifier should be generated by the the first involved LoWPAN node.

4.2. Flags

Some of the TCP flags are omitted because TCP control messages that set such flags (SYN, PUSH) are sent uncompressed. The uncompressed flags are : PUSH, FIN, Congestion Window Reduced (CWR) and ECN-Echo indication (ECE). These flags are present in the two bytes of the LOWPAN_TCPHC header.

4.3. Sequence and Acknowledgment numbers

The sequence number specifies the first data byte in the segment (except the first segment). The length of the sequence number field is four bytes.

In a TCP connection, the sequence number is incremented for each packet by a value between 0 and the MSS (Maximum Segment Size). Thus, the less significant bytes (LSB) are expected to change much frequently than the most significant bytes (MSB). Thus, it is often enough to send the N less-significant bytes if the 4-N bytes most significant bytes do not change. The decompressor module can deduce

the elided bytes from the previously received segments.

For example: The MSS value is 512 bytes and the current sequence number is 0x00f24512. Then, the sequence number of the next segment should be less or equal to 0x00f24712. The compressed segment can sent with only two bytes instead of sending all 4 bytes. The TCP sender can send two bytes 0x4712 and the TCP receiver should add the remaining static bytes 0x00f2. Using this method, we reduce to about 50% the length of sequence number or acknowledgment number fields if the MSS value does not exceed 65535 bytes.

The sequence number can be elided if a receiver is just acknowledging data segments and does not send data to the source (i.e., the receiver sends a pure TCP ACK). The same algorithm is used for the compression of the acknowledgment number and only bytes which are changed should be carried in-line. If the TCP sink does not generate data, the four bytes of the sequence number are omitted in all acknowledgment segments and only compressed acknowledgment fields should be sent.

4.4. Window

The window field can be omitted if it does not change in time. Moreover, only the window field bits that have been changed should be sent. The decompression deduces the value of this field from the last received full segment.

4.5. Urgent Pointer

The urgent pointer field is sent in full header format only if the urgent flag is set. Otherwise, this field is elided.

5. LOWPAN_TCPHC Packet Format

5.1. TCP segments types

Three types of packets are used in a TCP session with header compression:

Regular header TCP segment: A normal, uncompressed header. Does not carry any CID. Figure 7 shows the packet format of a regular TCP segment.

```
+-----+-----+-----+
|IPHC (NHC=0)|TCP header|Payload|
+-----+-----+-----+
```

Figure 7: Regular header TCP segment

Full header TCP segment: An uncompressed header that updates or refreshes the context for a packet stream. It carries a CID that will be used to identify the context. Figure 8 shows the packet format of a full header TCP segment.

```
+-----+-----+-----+-----+
|IPHC (NHC=1)|00000001|CID|TCP header|Payload|
+-----+-----+-----+-----+
```

Figure 8: Full header TCP segment

Compressed header TCP segment: Figure 9 shows the header stack of a compressed TCP segment.

```
+-----+-----+-----+-----+
|IPHC (NHC=1)|LOWPAN_TCPHC|CID|uncompressed TCP fields|Payload|
+-----+-----+-----+-----+
```

Figure 9: Compressed header TCP segment

5.2. LOWPAN_TCPHC Format

```

0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 | 1 | 0 | Id | Seq | Ack | W | CWR|ECE| F | P | T | S |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---
```

Figure 10: TCP Header Encoding

Id: Context Identifier Size

- 0: CID is coded in 8 bits.
- 1: CID is coded in 16 bits.

Seq: Sequence Number:

- 00: All 32 bits of Sequence Number are elided.
- 01: First 8 less-significant bits of Sequence Number are carried in-line. The remaining 24 bits are elided.
- 10: First 16 less-significant bits of Sequence Number are carried in-line. Last 16 bits of Sequence Number are elided.
- 11: All 32 bits of Sequence Number are carried in-line.

Ack: Acknowledgment Number:

- 00: All 32 bits of Acknowledgment Number are elided.
- 01: First 8 less-significant bits of Acknowledgment Number are carried in-line. The remaining 24 bits are elided.
- 10: First 16 less-significant bits of Acknowledgment Number are carried in-line. Last 16 bits of Acknowledgment Number are elided.

11: All 32 bits of Acknowledgment Number are carried in-line.
W: Window:
00: The Window field is elided.
01: The less-significant byte of Window field is carried in-line.
The most-significant byte is elided.
10: The most-significant byte of Window field is carried in-line.
The less-significant byte is elided.
11: Full 16 bits for Window field are carried in-line.
F: FIN flag.
P: PUSH flag.
CWR: Congestion Window Reduced flag.
ECE: ECN-Echo flag.
T: Set if the TCP header contains Timestamp option.
S: Set if the TCP header contains SACK option.

Fields carried in-line (in part or in whole) appear in the same order as they do in the TCP header format [RFC0793]. The TCP Length field must always be elided and it is inferred from lower layers using the 6LoWPAN fragmentation header or the MAC layer header.

5.3. Examples of compressed TCP headers

In this section, we present some examples of a compressed TCP header using LOWPAN_TCPHC.

5.3.1. Compressed header

Figure 11 represents a header of a TCP data segment whose window field has not changed from with respect its antecedent, the two bytes of the lowest bytes of the sequence number that have been changed. The size of this header is seven bytes.

3	1	2	2	2	1	1	1	1	1	1	1	1	8	16	16
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															
010 0 01 00 00 0 0 0 0 0 0 0 0 CID Seq.Number													Checksum		
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															

Figure 11: Compressed header TCP Header Encoding

6. TCP Option Compression

This section defines a compression method for the TCP options most likely to be used in 6LoWPAN. The "end of option" byte and the "no operation" byte are elided. Taking into account the characteristics of the LLNs, the "window scale" option are not supported because it is especially needed in broadband network and the window size in LLNs are limited to few segments. The "maximum segment size" option is

negotiated in the first control segments, thus they are not compressed. SACK option are not negotiated and are allowed by default. However, the ER can decide to allow or to deny an option sent in the SYN segment. LOWPAN_TCPHC compresses the mostly used TCP options : SACK and Timestamp. We assume that the SACK and Timestamp are not negotiated and used by default. LOWPAN_TCPHC specifies two bits for SACK and Timestamp TCP options. Figure 12 shows the structure of a TCP segment including options compressed using LOWPAN_TCPHC. MSS and SACK-Permitted options are sent in a SYN and they are compressed. The Window Scale Option (WSO) is useless in 6LoWPAN because it is more performance to use small windows than large windows. The size of the SACK option is 4 bytes and the size of Timestamp option is variable from 2 to 8 bytes.

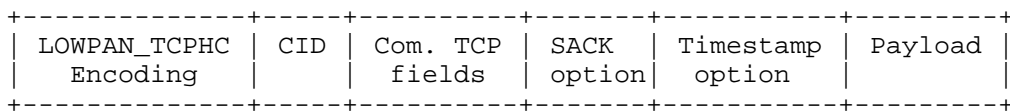


Figure 12: TCP Header Option Configuration

6.1. Selective Acknowledgment option

The TCP Selective Acknowledgment option (SACK) [RFC2018] [RFC2883] should be negotiated in set-up phase, then the option may be used when dropped segments are detected by the receiver. This option is to be used to convey extended acknowledgment information over an established connection. The left edge of the block can be replaced by the offset between the first byte of the segment and the right edge by the length of the block. The Left edge and the right edge will be coded in 16 bits.

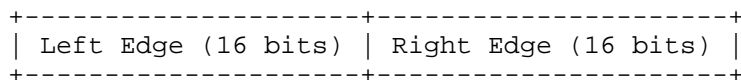


Figure 13: Compressed SACK option

6.2. Timestamp option

This option carries eight-byte timestamp fields. If timestamp options [RFC1323] are exchanged in the connection set-up phase, they are expected to appear on all subsequent segments. This overhead added by this option can be reduced: a TCP that does not send data is not interested to compute the RTTM. And thus, it can replay by sending only Timestamp Echo Reply field (TSecr). However, the Timestamp Value field (TSval) is more important for TCP that send data.

LOWPAN_TCPHC sends only bytes that have changed since the last segment to reduce the size of the Timestamp field. LOWPAN_TCPHC defines a bitmap field which specifies the bytes that are elided and the bytes that are carried in-line. Figure 14 shows the structure of the compressed TCP timestamp option fields.

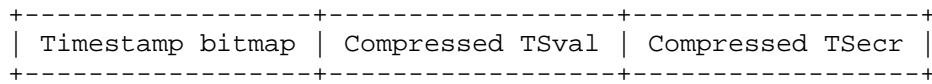


Figure 14: Compressed Timestamp option

7. Acknowledgments

This work has been funded by the Pole de Recherche Avancee en Communications (PRACom). The authors would like to thank Patrick Maille and Tiancong Zheng for their useful comments on an early version of this document.

8. References

[Perkins et al.]

Perkins, S. and M. Multa, "Dependency removal for transport protocol header compression over noisy channels", International Conference on Communications 1997, June 1997.

[Srivastava et al.]

Srivastava, A., Friday, R., Ritter, M., and W. San Filippo, "A study of TCP performance over wireless data networks", Vehicular Technology Conference, IEEE VTS 53rd, May 2001.

[Wang04]

Wang, R., "An experimental study of TCP/IP's Van Jacobson header compression behavior in lossy space environment", Vehicular Technology Conference, IEEE VTC 60th, September 2004.

[Medina05]

Medina, A., Allman, M., and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet", ACM SIGCOMM Computer Communications Review 35(2):37-51, April 2005.

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

Hui, J. and P. Thubert, "Compression Format for IPv6

Datagrams in 6LoWPAN Networks", October 2009.

[IEEE 802.15.4]

IEEE Computer Society, "IEEE Std. 802.15.4-2006", IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks- Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), October 2006.

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

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1144] Jacobson, V., "Compressing TCP/IP headers for low-speed serial links", RFC 1144, February 1990.

[RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

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

[RFC2507] Degermark, M., Nordgren, B., and S. Pink, "IP Header Compression", RFC 2507, February 1999.

[RFC2883] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An Extension to the Selective Acknowledgment (SACK) Option for TCP", RFC 2883, July 2000.

[RFC4413] West, M. and S. McCann, "TCP/IP Field Behavior", RFC 4413, March 2006.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,

"Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC4995] Jonsson, L-E., Pelletier, G., and K. Sandlund, "The RObust Header Compression (ROHC) Framework", RFC 4995, July 2007.

[RFC4996] Pelletier, G., Sandlund, K., Jonsson, L-E., and M. West, "RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)", RFC 4996, July 2007.

Authors' Addresses

Ahmed Ayadi
Telecom Bretagne
Rue de la Chataigneraie, CS 17607
35576 Cesson Sevigne cedex
France

Phone: +33 2 99 12 70 52
Email: ahmed.ayadi@telecom-bretagne.eu

David Ros
Telecom Bretagne
Rue de la Chataigneraie, CS 17607
35576 Cesson Sevigne cedex
France

Phone: +33 2 99 12 70 46
Email: david.ros@telecom-bretagne.eu

Laurent Toutain
Telecom Bretagne
Rue de la Chataigneraie, CS 17607
35576 Cesson Sevigne cedex
France

Phone: +33 2 99 12 70 26
Email: laurent.toutain@telecom-bretagne.eu

6LoWPAN Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2011

C. Bormann
Universitaet Bremen TZI
October 23, 2010

6LoWPAN Generic Compression of Headers and Header-like Payloads
draft-bormann-6lowpan-ghc-01

Abstract

This short I-D provides a complete design for a simple addition to 6LoWPAN Header Compression that enables the compression of generic headers and header-like payloads.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 26, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. The Header Compression Coupling Problem	3
1.1. Terminology	3
2. 6LoWPAN-GHC	4
3. Examples	5
4. Integrating 6LoWPAN-GHC into 6LoWPAN-HC	11
4.1. Compressing extension headers	11
5. IANA considerations	12
6. Acknowledgements	13
7. Security Considerations	14
8. References	15
8.1. Normative References	15
8.2. Informative References	15
Author's Address	16

1. The Header Compression Coupling Problem

[I-D.ietf-6lowpan-hc] defines a scheme for header compression in 6LoWPAN [RFC4944] packets. As with most header compression schemes, a new specification is needed for every new kind of header that needs to be compressed. In addition, [I-D.ietf-6lowpan-hc] does not define an extensibility scheme like the ROHC profiles defined in ROHC [RFC3095] [RFC5795]. This leads to the difficult situation that [I-D.ietf-6lowpan-hc] tends to be reopened and reexamined each time a new header receives consideration (or an old header is changed and reconsidered) in the 6lowpan/roll/core cluster of IETF working groups. At this rate, [I-D.ietf-6lowpan-hc] will never get completed (fortunately, by now it has passed WGLC, but the underlying problem remains unsolved).

The purpose of the present contribution is to plug into [I-D.ietf-6lowpan-hc] as is, using its NHC (next header compression) concept. We add a slightly less efficient, but vastly more general form of compression for headers of any kind and even for header-like payloads such as those exhibited by routing protocols, DHCP, etc. The objective is to arrive at something that can be defined on a single page and implemented in a couple of lines of code, as opposed to a general data compression scheme such as that defined in [RFC1951].

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP implementations.

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

2. 6LoWPAN-GHC

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

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

The code bytes are defined as follows:

code byte	Action	Argument
0kkkkkkk	Append k = 0b0kkkkkkk bytes of data in the bytecode argument (k < 96)	The k bytes of data
0110iiii	Append all bytes (possibly filling an incomplete byte with zero bits) from Context i	
0111iiii	Append 8 bytes from Context i; i.e., the context value truncated/extended to 8 bytes, and then append 0000 00FF FE00 (i.e., 14 bytes total)	
1000nnnn	Append 0b0000nnnn+2 bytes of zeroes	
1001nnnn	reserved	
101nssss	sa += 0b0ssss000, na += 0b0000n000	
11nnnkkk	n = na+0b00000nnn+2; s = 0b00000kkk+sa+n; append n bytes from previously output bytes, starting s bytes to the left of the current output pointer; set sa = 0, na = 0	

For the purposes of the backreferences, the expansion buffer is initialized with the pseudo-header as defined in [RFC2460], at the end of which the target buffer begins. These pseudo-header bytes are therefore available for backreferencing, but not copied into the final result.

3. Examples

This section demonstrates some relatively realistic examples derived from actual PCAP dumps taken at previous interops. Unfortunately, for these dumps, no context information was available, so the relatively powerful effect of context-based compression is not shown. (TBD: Add a couple DHCP examples.)

Figure 1 shows a quite short RPL control message that obviously cannot be improved much.

IP header:

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

Payload:

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

Pseudoheader:

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

copy: 04 9b 00 6b de

4 nulls: 82

Compressed:

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

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

Figure 1: A simple RPL example

Figure 2 shows a longer RPL control message that is improved a bit more (but would likely benefit additionally from a context reference). Note that the compressed output exposes an inefficiency in the simple-minded compressor used to generate it; this does not devalue the example since constrained nodes are quite likely to make use of simple-minded compressors.

```

IP header:
 60 00 00 00 00 5c 3a ff fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23 ff 02 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 1a
Payload:
 9b 01 7a 5f 00 f0 01 00 88 00 00 00 20 02 0d b8
 00 00 00 00 00 00 00 00 ff fe 00 fa ce 04 0e 00 14
 09 ff 00 00 01 00 00 00 00 00 00 00 08 1e 80 20
 ff ff ff ff ff ff ff ff 00 00 00 00 20 02 0d b8
 00 00 00 00 00 00 00 00 ff fe 00 fa ce 03 0e 40 00
 ff ff ff ff 20 02 0d b8 00 00 00 00
Pseudoheader:
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
 00 00 00 5c 00 00 00 3a
copy: 09 9b 01 7a 5f 00 f0 01 00 88
3 nulls: 81
copy: 04 20 02 0d b8
7 nulls: 85
ref(52): ff fe 00 -> ref 10lnssss 0 6/1lnnnkkk 1 1: a6 c9
copy: 08 fa ce 04 0e 00 14 09 ff
2 nulls: 80
copy: 01 01
7 nulls: 85
copy: 06 08 1e 80 20 ff ff
ref(2): ff ff -> ref 1lnnnkkk 0 0: c0
ref(4): ff ff ff ff -> ref 1lnnnkkk 2 0: d0
4 nulls: 82
ref(48): 20 02 0d b8 00 00 00 00 00 00 00 00 ff fe 00 fa ce
-> ref 10lnssss 1 4/1lnnnkkk 6 0: b4 f0
copy: 03 03 0e 40
ref(9): 00 ff -> ref 1lnnnkkk 0 7: c7
ref(28): ff ff ff -> ref 10lnssss 0 3/1lnnnkkk 1 1: a3 c9
ref(24): 20 02 0d b8 00 00 00 00
-> ref 10lnssss 0 2/1lnnnkkk 6 0: a2 f0
Compressed:
 09 9b 01 7a 5f 00 f0 01 00 88 81 04 20 02 0d b8
 85 a6 c9 08 fa ce 04 0e 00 14 09 ff 80 01 01 85
 06 08 1e 80 20 ff ff c0 d0 82 b4 f0 03 03 0e 40
 c7 a3 c9 a2 f0
Was 92 bytes; compressed to 53 bytes, compression factor 1.74

```

Figure 2: A longer RPL example

Figure 3 shows an the effect of compressing a simple ND neighbor solicitation (again, no context-based compression).

IP header:

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

Payload:

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

Pseudoheader:

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

copy: 04 87 00 a7 68

4 nulls: 82

ref(32): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23

-> ref 10lnssss 1 2/1lnnnkkk 6 0: b2 f0

copy: 04 01 01 3b d3

4 nulls: 82

copy: 02 1f 02

5 nulls: 83

copy: 02 06 00

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

copy: 02 20 24

Compressed:

```
04 87 00 a7 68 82 b2 f0 04 01 01 3b d3 82 02 1f
02 83 02 06 00 a2 db 02 20 24
```

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

Figure 3: An ND neighbor solicitation

Figure 4 shows the compression of an ND neighbor advertisement.

IP header:

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

Payload:

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

Pseudoheader:

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

copy: 05 88 00 26 6c c0

3 nulls: 81

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

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

copy: 04 02 01 fa ce

4 nulls: 82

copy: 02 1f 02

5 nulls: 83

copy: 02 06 00

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

copy: 02 20 24

Compressed:

```
05 88 00 26 6c c0 81 b4 f0 04 02 01 fa ce 82 02
1f 02 83 02 06 00 a2 db 02 20 24
```

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

Figure 4: An ND neighbor advertisement

Figure 5 shows the compression of an ND router solicitation. Note that the relatively good compression is not caused by the many zero bytes in the link-layer address of this particular capture (which are unlikely to occur in practice): 7 of these 8 bytes are copied from the pseudo header (the 8th byte cannot be copied as the universal/local bit needs to be inverted).

IP header:

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

Payload:

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

Pseudoheader:

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

copy: 04 85 00 90 65

ref(33): 00 00 00 00 01 -> ref 10lnssss 0 3/1lnnnkkk 3 4: a3 dc

copy: 02 02 ac

ref(42): de 48 00 00 00 00 01

-> ref 10lnssss 0 4/1lnnnkkk 5 3: a4 eb

6 nulls: 84

Compressed:

```
04 85 00 90 65 a3 dc 02 02 ac a4 eb 84
```

Was 24 bytes; compressed to 13 bytes, compression factor 1.85

Figure 5

Figure 6 shows the compression of an ND router advertisement. The indefinite lifetime is compressed to four bytes by backreferencing; this could be improved (at the cost of minor additional decompressor complexity) by including some simple runlength mechanism.

```

IP header:
 60 00 00 00 00 60 3a ff fe 80 00 00 00 00 00 00
 10 34 00 ff fe 00 11 22 fe 80 00 00 00 00 00 00
 ae de 48 00 00 00 00 01
Payload:
 86 00 55 c9 40 00 0f a0 1c 5a 38 17 00 00 07 d0
 01 01 11 22 00 00 00 00 03 04 40 40 ff ff ff ff
 ff ff ff ff 00 00 00 00 20 02 0d b8 00 00 00 00
 00 00 00 00 00 00 00 00 20 02 40 10 00 00 03 e8
 20 02 0d b8 00 00 00 00 21 03 00 01 00 00 00 00
 20 02 0d b8 00 00 00 00 00 00 00 00 ff fe 00 11 22
Pseudoheader:
 fe 80 00 00 00 00 00 00 10 34 00 ff fe 00 11 22
 fe 80 00 00 00 00 00 00 ae de 48 00 00 00 00 01
 00 00 00 60 00 00 00 3a
copy: 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17
2 nulls: 80
copy: 06 07 d0 01 01 11 22
4 nulls: 82
copy: 06 03 04 40 40 ff ff
ref(2): ff ff -> ref 11nnnkkk 0 0: c0
ref(4): ff ff ff ff -> ref 11nnnkkk 2 0: d0
4 nulls: 82
copy: 04 20 02 0d b8
12 nulls: 8a
copy: 04 20 02 40 10
ref(38): 00 00 03 -> ref 10lnssss 0 4/11nnnkkk 1 3: a4 cb
copy: 01 e8
ref(24): 20 02 0d b8 00 00 00 00
-> ref 10lnssss 0 2/11nnnkkk 6 0: a2 f0
copy: 02 21 03
ref(84): 00 01 00 00 00 -> ref 10lnssss 0 9/11nnnkkk 3 7: a9 df
ref(40): 00 20 02 0d b8 00 00 00 00 00 00 00
-> ref 10lnssss 1 3/11nnnkkk 2 4: b3 d4
ref(120): ff fe 00 11 22
-> ref 10lnssss 0 14/11nnnkkk 3 3: ae db
Compressed:
 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17 80 06 07
 d0 01 01 11 22 82 06 03 04 40 40 ff ff c0 d0 82
 04 20 02 0d b8 8a 04 20 02 40 10 a4 cb 01 e8 a2
 f0 02 21 03 a9 df b3 d4 ae db
Was 96 bytes; compressed to 58 bytes, compression factor 1.66

```

Figure 6: An ND router advertisement

4. Integrating 6LoWPAN-GHC into 6LoWPAN-HC

6LoWPAN-GHC is intended to plug in as an NHC format for 6LoWPAN-HC [I-D.ietf-6lowpan-hc]. This section shows how this can be done (without supplying the detailed normative text yet, although it could be implemented from this page).

GHC is by definition generic and can be applied to different kinds of packets. All the examples given above are for ICMPv6 packets; it is trivial to define an NHC format for ICMPv6 based on GHC.

In addition it may be useful to include an NHC format for UDP, as many headerlike payloads (e.g., DHCPv6) are carried in UDP. [I-D.ietf-6lowpan-hc] already defines an NHC format for UDP (11110CPP). What remains to be done is to define an analogous NHC byte formatted, e.g. as shown in Figure 7, and simply reference the existing specification, indicating that for 0b11010cpp NHC bytes, the UDP payload is not supplied literally but compressed by 6LoWPAN-GHC.

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

Figure 7: A possible NHC byte for UDP GHC

To stay in the same general numbering space, we propose 0b11011111 as the NHC byte for IPCMPv6 GHC.

4.1. Compressing extension headers

If the compression of specific extension headers is considered desirable, this can be added in a similar way, e.g. as in Figure 8 (however, probably only EID 0 to 3 need to be assigned). As there is no easy way to extract the length field from the GHC-encoded header before decoding, this would make detecting the end of the extension header somewhat complex. The easiest (and most efficient) approach is to completely elide the length field (in the same way NHC already elides the next header field in certain cases) and reconstruct it only on decompression. Instead, the reserved bytecode 0b10010000 would be assigned as a stop marker.

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

Figure 8: A possible NHC byte for extension header GHC

5. IANA considerations

In the IANA registry for the LOWPAN_NHC header type, IANA would need to add the assignments in Figure 9.

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

Figure 9: IANA assignments for the NHC byte

*) if the functionality of Section 4.1 is made part of this document.

6. Acknowledgements

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

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

7. Security Considerations

(To be worked out. Probably mostly about the need to avoid buffer overflows and out-of-area references during decompression.)

8. References

8.1. Normative References

- [I-D.ietf-6lowpan-hc]
Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams in 6LoWPAN Networks", draft-ietf-6lowpan-hc-13 (work in progress), September 2010.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

8.2. Informative References

- [I-D.oflynn-6lowpan-icmphc]
O'Flynn, C., "ICMPv6/ND Compression for 6LoWPAN Networks", draft-oflynn-6lowpan-icmphc-00 (work in progress), July 2010.
- [RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.
- [RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, March 2010.

Author's Address

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

Phone: +49-421-218-63921
Fax: +49-421-218-7000
Email: cabo@tzi.org

6lowpan Working Group
Internet-Draft
Updates: 4944 (if approved)
Intended status: Standards Track
Expires: April 29, 2011

Z. Shelby, Ed.
Sensinode
S. Chakrabarti
IP Infusion
E. Nordmark
Oracle, Inc.
October 26, 2010

Neighbor Discovery Optimization for Low-power and Lossy Networks
draft-ietf-6lowpan-nd-14

Abstract

The IETF 6LoWPAN working group defines IPv6 over Low-power Wireless Personal Area Networks such as IEEE 802.15.4. This and other similar link technologies have limited or no usage of multicast signaling due to energy conservation. In addition, the wireless network may not strictly follow traditional concept of IP subnets and IP links. IPv6 Neighbor Discovery was not designed for non-transitive wireless links. The traditional IPv6 link concept and heavy use of multicast make the protocol inefficient and sometimes impractical in a low power and lossy network. This document describes simple optimizations to IPv6 Neighbor Discovery, addressing mechanisms and duplicate address detection for 6LoWPAN and similar networks.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 29, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
1.1. The Shortcomings of IPv6 Neighbor Discovery	5
1.2. Mesh-under and Route-over Concepts	6
1.3. Applicability, Goals and Assumptions	7
1.4. Optional Features	9
2. Terminology	9
3. Protocol Overview	10
3.1. Extensions to RFC4861	11
3.2. Address Assignment	12
3.3. Host-to-Router Interaction	12
3.4. Router-to-Router Interaction	13
3.5. Neighbor Cache Management	14
4. New Neighbor Discovery Options and Messages	15
4.1. Address Registration Option	15
4.2. 6LoWPAN Context Option	17
4.3. Authoritative Border Router Option	18
4.4. Duplicate Address messages	19
5. Host Behavior	21
5.1. Forbidden Actions	21
5.2. Interface Initialization	21
5.3. Sending a Router Solicitation	22
5.4. Processing a Router Advertisement	22
5.4.1. Address configuration	22
5.4.2. Storing Contexts	23
5.4.3. Maintaining Prefix and Context Information	23
5.5. Registration and Neighbor Unreachability Detection	24
5.5.1. Sending a Neighbor Solicitation	24
5.5.2. Processing a Neighbor Advertisement	24
5.5.3. Recovering from Failures	25
5.6. Next-hop Determination	25
5.7. Address Resolution	26
5.8. Sleeping	26
5.8.1. Picking an Appropriate Registration Lifetime	26
5.8.2. Behavior on Wakeup	27
6. Router Behavior for 6LR and 6LBR	27

6.1.	Forbidden Actions	27
6.2.	Interface Initialization	28
6.3.	Processing a Router Solicitation	28
6.4.	Periodic Router Advertisements	29
6.5.	Processing a Neighbor Solicitation	29
6.5.1.	Checking for Duplicates	29
6.5.2.	Returning Address Registration Errors	30
6.5.3.	Updating the Neighbor Cache	30
6.5.4.	Next-hop Determination	30
6.5.5.	Address Resolution between Routers	31
7.	Border Router Behavior	31
7.1.	Prefix Determination	32
7.2.	Context Configuration and Management	32
8.	Optional Behavior	33
8.1.	Multihop Prefix and Context Distribution	33
8.1.1.	6LBRs Sending Router Advertisements	33
8.1.2.	Routers Sending Router Solicitations	34
8.1.3.	Routers Processing Router Advertisements	34
8.1.4.	Storing the Information	35
8.1.5.	Sending Router Advertisements	35
8.2.	Multihop Duplicate Address Detection	36
8.2.1.	Message Validation for DAR and DAC	37
8.2.2.	Conceptual Data Structures	38
8.2.3.	6LR Sending a Duplicate Address Request	38
8.2.4.	6LBR Receiving a Duplicate Address Request	39
8.2.5.	Processing a Duplicate Address Confirmation	39
8.2.6.	Recovering from Failures	39
9.	Protocol Constants	40
10.	Examples	40
10.1.	Message Examples	41
10.2.	Host Bootstrapping Example	42
10.2.1.	Host Bootstrapping Messages	43
10.3.	Router Interaction Example	46
10.3.1.	Bootstrapping a Router	46
10.3.2.	Updating the Neighbor Cache	46
11.	Security Considerations	47
12.	IANA Considerations	47
13.	Guideline for New Features	48
14.	Acknowledgments	51
15.	Changelog	51
16.	References	57
16.1.	Normative References	57
16.2.	Informative References	58
	Authors' Addresses	59

1. Introduction

The IPv6-over-IEEE 802.15.4 [RFC4944] document specifies how IPv6 is carried over an IEEE 802.15.4 network with the help of an adaptation layer which sits between the MAC layer and the IP network layer. A link in a LoWPAN is characterized as lossy, low-power, low bit-rate, short range, with many nodes saving energy with long sleep periods. Multicast as used in IPv6 Neighbor Discovery [RFC4861] is not desirable in such a wireless low-power and lossy network. Moreover, LoWPAN links are asymmetric and non-transitive in nature. A LoWPAN is potentially composed of a large number of overlapping radio ranges. Although a given radio range has broadcast capabilities, the aggregation of these is a complex Non-Broadcast MultiAccess (NBMA, [RFC2491]) structure with generally no LoWPAN-wide multicast capabilities. Link-local scope is in reality defined by reachability and radio strength. Thus we can consider a LoWPAN to be made up of links with undetermined connectivity properties as in [RFC5889], along with the corresponding address model assumptions defined therein.

This specification introduces the following optimizations to IPv6 Neighbor Discovery [RFC4861] specifically aimed at low-power and lossy networks such as LoWPANs:

- o Host-initiated interactions to allow for sleeping hosts.
- o Elimination of multicast-based address resolution for hosts.
- o Elimination of redirects since they are problematic on links with non-transitive connectivity.
- o A host address registration feature using a new option in unicast Neighbor Solicitation and Neighbor Advertisement messages.
- o A new Neighbor Discovery option to distribute 6LoWPAN header compression context to hosts.
- o Optional multihop distribution of prefix and 6LoWPAN header compression context.
- o Optional multihop duplicate address detection which uses two new ICMPv6 message types.

The document defines three new ICMPv6 message options: the required Address Registration option and the optional Authoritative Border Router and 6LoWPAN Context options. It also defines two new ICMPv6 message types: the Duplicate Address Request and Duplicate Address Confirmation.

1.1. The Shortcomings of IPv6 Neighbor Discovery

IPv6 Neighbor Discovery [RFC4861] provides several important mechanisms used for Router Discovery, Address Resolution, Duplicate Address Detection, Redirect, along with Prefix and Parameter Discovery.

Following power-on and initialization of the network in IPv6 Ethernet networks, a node joins the solicited-node multicast address on the interface and then performs Duplicate Address Detection (DAD) for the acquired link-local address by sending a solicited-node multicast message to the link. After that it sends multicast messages to the all-router address to solicit router advertisements. If the host receives a valid Router Advertisement with the "A" flag, it autoconfigures the IPv6 address with the advertised prefix in the Router Advertisement (RA) message. Besides this, the IPv6 routers usually send router advertisements periodically on the network. RAs are sent to the all-node multicast address. Nodes send Neighbor Solicitation/Neighbor Advertisement messages to resolve the IPv6 address of the destination on the link. The Neighbor Solicitation messages used for address resolution are multicast. The Duplicate Address Detection procedure and the use of periodic Router Advertisement messages assumes that the nodes are powered on and reachable most of the time.

In Neighbor Discovery the routers find the hosts by assuming that a subnet prefix maps to one broadcast domain, and then multicast Neighbor Solicitation messages to find the host and its link-layer address. Furthermore, the DAD use of multicast assumes that all hosts that autoconfigure IPv6 addresses from the same prefix can be reached using link-local multicast messages.

Note that the 'L' (on-link) bit in the Prefix Information option can be set to zero in Neighbor Discovery, which makes the host not use multicast Neighbor Solicitation (NS) messages for address resolution of other hosts, but routers still use multicast NS messages to find the hosts.

In a LoWPAN, primarily two types of network topologies are found - star networks and mesh networks. A star network is similar to a regular IPv6 subnet with a router and a set of nodes connected to it via the same non-transitive link. But in Mesh networks, the nodes are capable of routing and forwarding packets. Due to the lossy nature of wireless communication and a changing radio environment, the IPv6-link node-set may change due to external physical factors. Thus the link is often unstable and the nodes appear to be moving without necessarily moving physically.

A LoWPAN can use two types of link-layer addresses; 16-bit short addresses and 64-bit unique addresses as defined in [RFC4944]. Moreover, the available link-layer payload size is on the order of less than 100 bytes thus header compression is very useful.

Considering the above characteristics in a LoWPAN, and the IPv6 Neighbor Discovery [RFC4861] protocol design center, some optimizations and extensions to Neighbor Discovery are useful for the wide deployment of IPv6 over low-powered and lossy networks such as 6LoWPANs.

1.2. Mesh-under and Route-over Concepts

In the 6LoWPAN context, often a link-layer mesh routing mechanism is referred to as "mesh-under" while routing/forwarding packets using IP-layer addresses is referred to as "route-over". The difference between mesh-under and route-over is similar to a bridged-network versus IP-routing using Ethernet. In a mesh-under network all nodes are on the same link which is served by one or more routers, which we call 6LoWPAN Border Routers (6LBR). In a route-over network, there are multiple links in the 6LoWPAN. Unlike fixed IP links, these link's members may be changing due to the nature of the low-power and lossy behavior of wireless technology. Thus a route-over network is made up of a flexible set of links interconnected by interior routers, which we call 6LoWPAN Routers (6LR).

This specification is applicable to both mesh-under and route-over networks. However, in route-over networks, we have two types of routers - 6LBRs and 6LRs. 6LoWPAN Border Routers sit at the boundary of the 6LoWPAN and the rest of the network while 6LoWPAN Routers are inside the LoWPAN. 6LoWPAN Routers are assumed to be running a routing protocol.

In a mesh-under configuration a 6LBR is acting as the IPv6 router where all the hosts in the LoWPAN are on the same link, thus they are only one IP hop away. No 6LoWPAN Routers exist in this topology as forwarding is handled by a link-layer mesh routing protocol.

In a route-over configuration, Neighbor Discovery operations take place between hosts and 6LRs or 6LBRs. The 6LR nodes are able to send and receive Router Advertisements, Router Solicitations as well as forward and route IPv6 packets. Here packet forwarding happens at the IP layer.

In both types of configurations, hosts do not take part in routing and forwarding packets and they act as simple IPv6 hosts.

1.3. Applicability, Goals and Assumptions

This document specifies a set of behaviors between hosts and routers. An implementation that adheres to this document **MUST** implement those behaviors. The document also specifies a set of behaviors (multihop prefix or context dissemination, and separately multihop duplicate address detection) which are **OPTIONAL** to use. An implementation of this specification **SHOULD** implement those optional to use pieces.

The optimizations described in this document apply to different topologies. They are most useful for route-over and mesh-under configurations in Mesh topologies. However, Star topology configurations will also benefit from the optimizations due to minimized signaling, robust handling of the non-transitive link, and header compression context information.

The document has the following main goals and assumptions.

Goals:

- o Optimize Neighbor Discovery with a mechanism that is minimal yet sufficient for the operation in both mesh-under and route-over configurations.
- o Make the host to router interaction the same whether mesh-under or route-over is used.
- o Minimize signaling by avoiding the use of multicast flooding and reducing the use of link-scope multicast messages.
- o Optimize the interfaces between hosts and their default routers.
- o Support for sleeping hosts.
- o Minimize the complexity of nodes.
- o Disseminate context information to hosts as needed by [I-D.ietf-6lowpan-hc].
- o Optionally disseminate context information and prefix information from the border to all routers in a LoWPAN.
- o Optional duplicate address detection mechanism suitable for route-over LoWPANs.

Assumptions:

- o EUI-64 addresses are globally unique.
- o All nodes in the network have an EUI-64 interface identifier in order to do address auto-configuration and detect duplicate addresses.
- o The link layer technology is assumed to be low-power and lossy, exhibiting undetermined connectivity, such as IEEE 802.15.4 [RFC4944]. However, the Address Registration mechanism might be useful for other link layer technologies.
- o A 6LoWPAN is configured to share one or more global IPv6 address prefixes to enable hosts to move between routers in the 6LoWPAN without changing their IPv6 addresses.
- o When using the optional DAD mechanism of Section 8.2 it is assumed that 6LRs register with all the 6LBRs.
- o If IEEE 802.15.4 16-bit short addresses are used, then some technique is used to ensure uniqueness of those link-layer addresses. That could be done using DHCPv6, the Address Registration Option based duplicate address detection (specified in Section 8.2) or other techniques outside of the scope of this document.
- o In order to preserve the uniqueness of addresses not derived from an EUI-64, they must be either assigned or checked for duplicates in the same way throughout the LoWPAN. This can be done using DHCPv6 for assignment and/or using the duplicate address detection mechanism specified in Section 8.2 (or any other protocols developed for that purpose).
- o In order for [I-D.ietf-6lowpan-hc] to operate correctly, the compression context must match for all the hosts, 6LRs, and 6LBRs that can send, receive, or forward a given packet. If Section 8.1 is used to distribute context information this implies that all the 6LBRs must coordinate the context information they distribute within a single 6LoWPAN.
- o This specification describes the operation of ND within a single LoWPAN. The participation of a node in multiple LoWPANs simultaneously may be possible, but is out of scope of this document.
- o Since the 6lowpan shares one single prefix throughout the network, mobility of nodes within the lowpan is transparent. Inter-lowpan mobility is out-of-scope of this document. When a node moves away from the registered default router, it SHOULD first de-register by

sending a registration message with zero lifetime and then registers with a new default router that is reachable. If the node loses connectivity with the default router involunterily, then the default router does not know the node has moved until it runs the unreachability detection. In order to optimize the time for detecting unreachability of a run-away node, a default-router may use link-layer indication or configure a lower NCE life-time value and low registration life-time value - so that it can remove the NCE of the run-away node as soon as possible.

1.4. Optional Features

This document defines the optimization of Neighbor Discovery messages in host-router interface and introduces the communication in case of Route-over topology. The multihop prefix distribution by the 6LBR and multihop Duplicate Address Detection mechanisms, as well as 6LoWPAN context option are optional features for a 6LoWPAN deployment. A guideline for feature implementation and deployment is provided at the end of the document.

2. Terminology

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

This specification requires readers to be familiar with all the terms and concepts that are discussed in "Neighbor Discovery for IP version 6" [RFC4861] "IPv6 Stateless Address Autoconfiguration" [RFC4862], "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] and "IP Addressing Model in Ad Hoc Networks" [RFC5889].

This specification makes extensive use of the same terminology defined in [RFC4861] unless otherwise defined below.

6LoWPAN link:

A wireless link determined by single IP hop reachability of neighboring nodes. These are considered links with undetermined connectivity properties as in [RFC5889].

6LoWPAN Node (6LN):

A 6LoWPAN Node is any host or router participating in a LoWPAN. This term is used when referring to situations in which either a host or router can play the role described.

6LoWPAN Router (6LR):

An intermediate router in the LoWPAN who can communicate with other 6LoWPAN routers in the same LoWPAN. 6LoWPAN routers are present only in route-over topologies.

6LoWPAN Border Router (6LBR):

A border router located at the junction of separate 6LoWPAN networks or between a 6LoWPAN network and another IP network. There may be one or more 6LBRs at the 6LoWPAN network boundary. A 6LBR is the responsible authority for IPv6 Prefix propagation for the 6LoWPAN network it is serving. An isolated LoWPAN also contains a 6LBR in the network, which provides the prefix(es) for the isolated network.

Router:

Either a 6LR or a 6LBR. Note that nothing in this document precludes a node being a router on some interfaces and a host on other interfaces as allowed by [RFC2460].

Mesh-under:

A topology where hosts are connected to a 6LBR through a mesh using link-layer forwarding. Thus in a mesh-under configuration all IPv6 hosts in a LoWPAN are only one IP hop away from the 6LBR. This topology simulates the typical IP-subnet topology with one router with multiple nodes in the same subnet.

Route-over:

A topology where hosts are connected to the 6LBR through the use of intermediate layer-3 (IP) routing. Here hosts are typically multiple IP hops away from a 6LBR. The route-over topology typically consists of a 6LBR, a set of 6LRs and hosts.

Registration:

The process during which a LoWPAN node sends an Neighbor Solicitation message with an Address Registration option to a Router creating a Neighbor Cache entry for the LoWPAN node with a specific timeout. Thus for 6LoWPAN Routers the Neighbor Cache doesn't behave like a cache. Instead it behaves as a registry of all the host addresses that are attached to the Router.

3. Protocol Overview

These Neighbor Discovery optimizations are applicable to both mesh-under and route-over configurations. In a mesh-under configuration only 6LoWPAN Border Routers and hosts exist; there are no 6LoWPAN routers in mesh-under topologies.

The most important part of the optimizations is the evolved host-to-router interaction that allows for sleeping nodes and avoids using multicast Neighbor Discovery messages except for the case of a host finding an initial set of default routers, and redoing such determination when that set of routers have become unreachable.

The protocol also provides for header compression [I-D.ietf-6lowpan-hc] by carrying header compression information in a new option in Router Advertisement messages.

In addition, there are optional and separate mechanisms that can be used between 6LRs and 6LBRs to perform multihop Duplicate Address Detection and distribution of the Prefix and compression Context information from the 6LBRs to all the 6LRs, which in turn use normal Neighbor Discovery mechanisms to convey this information to the hosts.

The protocol is designed so that the host-to-router interaction is not affected by the configuration of the 6LoWPAN; the host-to-router interaction is the same in a mesh-under and route-over configuration.

3.1. Extensions to RFC4861

This document specifies the following optimizations and extensions to IPv6 Neighbor Discovery [RFC4861]:

- o Host initiated refresh of Router Advertisement information. This removes the need for periodic or unsolicited Router Advertisements from routers to hosts.
- o No Duplicate Address Detection (DAD) is performed if EUI-64 based IPv6 addresses are used (as these addresses are assumed to be globally unique).
- o DAD is optional if DHCPv6 is used to assign addresses.
- o A New Address Registration mechanism using new Address Registration option between hosts and routers. This removes the need for Routers to use multicast Neighbor Solicitations to find hosts, and supports sleeping hosts. This also enables the same IPv6 address prefix(es) to be used across a route-over 6LoWPAN. It provides the host-to-router interface for Duplicate Address Detection.
- o A new optional Router Advertisement option for Context information used by 6LoWPAN header compression.

- o A new optional mechanism to perform Duplicate Address Detection across a route-over 6LoWPAN using the new Duplicate Address Request and Confirmation messages.
- o New optional mechanisms to distribute Prefixes and Context information across a route-over network which uses a new Authoritative Border Router option to control the flooding of configuration changes.
- o A few new default protocol constants are introduced and some existing Neighbor Discovery protocol constants are tuned.

3.2. Address Assignment

Hosts in a 6LoWPAN configure their IPv6 address as specified in [RFC4861] and [RFC4862] based on the information received in Router Advertisement messages. The use of the M flag in this optimization is however more restrictive than in [RFC4861]. When the M flag is set a host is required to use DHCPv6 to assign any non-EUI-64 addresses. When the M flag is not set, the LoWPAN is required to support duplicate address detection, thus a host can then safely use the address registration mechanism to check non-EUI-64 addresses for uniqueness.

6LRs MAY use the same mechanisms to configure their IPv6 addresses.

The 6LBRs are responsible for managing the prefix(es) assigned to the 6LoWPAN, using manual configuration, DHCPv6 Prefix Delegation [RFC3633], or other mechanisms. In an isolated LoWPAN a ULA [RFC4193] prefix SHOULD be generated by the 6LBR.

3.3. Host-to-Router Interaction

A host sends Router Solicitation messages at startup and also when it suspects that one of its default routers has become unreachable (after Neighbor Unreachability Detection towards the router fails).

Hosts receive Router Advertisement messages typically containing the Authoritative Border Router option (ABRO) and may optionally contain one or more 6LoWPAN Context options (6CO) in addition to the existing Prefix Information options (PIO) as described in [RFC4861].

When a host has configured a non-link-local IPv6 address, it registers that address with one or more of its default routers using the Address Registration option (ARO) in an NS message. The host chooses a lifetime of the registration and repeats the ARO option periodically (before the lifetime runs out) to maintain the registration. The lifetime should be chosen in such a way as to

maintain the registration even while a host is sleeping. Likewise, mobile nodes that change their point of attachment often, should use a suitably short lifetime.

The registration can fail (an ARO option returned to the host with a non-zero Status) if the router determines that the IPv6 address is already used by another host, that is, is used by a host with a different EUI-64. This can be used to support non-EUI-64 based addresses such as temporary IPv6 addresses [RFC4941] or addresses based on an Interface ID that is a IEEE 802.15.4 16-bit short addresses. Failure can also occur if the Neighbor Cache on that router is full.

The re-registration of an address can be combined with Neighbor Unreachability Detection (NUD) of the router since both use unicast Neighbor Solicitation messages. This makes things efficient when a host wakes up to send a packet and both need to perform NUD to check that the router is still reachable, and refresh its registration with the router.

The response to an address registration might not be immediate since in route-over configurations the 6LR might perform Duplicate Address Detection against the 6LBR. A host retransmits the Address Registration option until it is acknowledged by the receipt of a Address Registration option.

As part of the optimizations, Address Resolution is not performed by multicasting Neighbor Solicitation messages as in [RFC4861]. Instead, the routers maintain Neighbor Cache entries for all registered IPv6 addresses. If the address is not in the Neighbor Cache in the router, then the address either doesn't exist, or is assigned to a host attached to some other router in the 6LoWPAN, or is external to the 6LoWPAN. In a route-over configuration the routing protocol is used to route such packets toward the destination.

3.4. Router-to-Router Interaction

The optional new router-to-router interaction is only for the route-over configuration where 6LRs are present. It is optional in this protocol since the functions it provides might be better provided by other protocol mechanisms, be it DHCPv6, link-layer mechanisms, the routing protocol, or something else. It is however assumed that all 6LRs in a network are configured to perform these functions homogeneously. Some mechanisms from this protocol might be used for router-to-router interaction, while others are provided by other protocols. For instance, context information and/or prefix information might be disseminated using this protocol, while

Duplicate Address Detection is done using some other protocol.

6LRs MAY act like a host during system startup and prefix configuration by sending Router Solicitation messages and autoconfiguring their IPv6 addresses unlike routers in [RFC4861].

When multihop prefix or context dissemination is used then the 6LRs store the ABRO, 6CO and Prefix Information received (directly or indirectly) from the 6LBRs and redistribute this information in the Router Advertisement they send to other 6LRs or send to hosts in response to a Router Solicitations. There is a version number field in the ABRO which is used to limit the flooding of updated information between the 6LRs.

Optionally the 6LRs can perform Duplicate Address Detection against one or more 6LBRs using the new Duplicate Address Request (DAR) and Confirmation (DAC) messages, which carry the information from the Address Registration option. The DAR and DAC messages will be forwarded between the 6LR and 6LBRs thus the [RFC4861] rule for checking hop limit=255 does not apply to the DAR and DAC messages. Those multihop DAD messages MUST NOT modify any Neighbor Cache entries on the routers since we do not have the security benefits provided by the hop limit=255 check.

3.5. Neighbor Cache Management

The use of explicit registrations with lifetimes plus the desire to not multicast Neighbor Solicitation messages for hosts imply that we manage the Neighbor Cache entries slightly differently than in [RFC4861]. This results in three different types of NCEs and the types specify how those entries can be removed:

- Garbage-collectible: Entries that are subject to the normal rules in [RFC4861] that allow for garbage collection when low on memory.
- Registered: Entries that have an explicit registered lifetime and are kept until this lifetime expires or they are explicitly unregistered.
- Tentative: Entries that are temporary with a short lifetime, which typically get converted to Registered entries.

Note that the type of the NCE is orthogonal to the states specified in [RFC4861].

When a host interacts with a router by sending Router Solicitations

this results in a Tentative NCE. Once a node successfully registers with a Router the result is a Registered NCE. As Routers send RAs to hosts, and when routers optionally receive RA messages or receive multicast NS messages from other Routers the result is Garbage-collectible NCEs. There can only be one kind of NCE for an IP address at a time.

Neighbor Cache entries on Routers can additionally be added or deleted by a routing protocol used in the 6LoWPAN. This is useful if the routing protocol carries the link-layer addresses of the neighboring routers. Depending on the details of such routing protocols such NCEs could be either Registered or Garbage-collectible.

4. New Neighbor Discovery Options and Messages

This section defines new Neighbor Discovery message options used by this specification. The Address Registration Option is mandatory, whereas the Authoritative Border Router Option and 6LoWPAN Context Option are optional. This section also defines the optional and new Duplicate Address Request and Confirmation messages.

4.1. Address Registration Option

The routers need to know the set of host IP addresses that are directly reachable and their corresponding link-layer addresses. This needs to be maintained as the radio reachability changes. For this purpose an Address Registration Option (ARO) is introduced, which can be included in unicast Neighbor Solicitation (NS) messages sent by hosts. Thus it can be included in the unicast NS messages that a host sends as part of Neighbor Unreachability Detection to determine that it can still reach a default router. The ARO is used by the receiving router to reliably maintain its Neighbor Cache. The same option is included in corresponding Neighbor Advertisement (NA) messages with a Status field indicating the success or failure of the registration. This option is always host initiated.

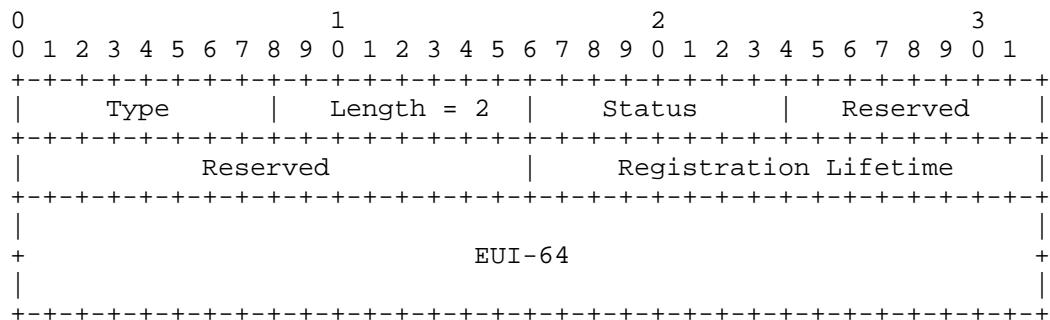
The information contained in the ARO is also included in optional multihop DAR and DAC messages used between 6LRs to 6LBRs, but the option itself is not used in those messages.

The ARO is required for reliability and power saving. The lifetime field provides flexibility to the host to register an address which should be usable (continue to be advertised by the 6LR in the routing protocol etc.) during its intended sleep schedule.

The sender of the NS also includes the EUI-64 [EUI64] of the

interface it is registering an address from. This is used as a unique ID for the detection of duplicate addresses. It is used to tell the difference between the same node re-registering its address and a different node (with a different EUI-64) registering an address that is already in use by someone else. The EUI-64 is also used to deliver an NA carrying an error Status code to the EUI-64 based link-local IPv6 address of the host (see Section 6.5.2).

When the ARO is used by hosts an SLLA option MUST be included and the address that is to be registered MUST be the IPv6 source address of the Neighbor Solicitation message.



Fields:

Type: TBD1

Length: 8-bit unsigned integer. The length of the option in units of 8 bytes. Always 2.

Status: 8-bit unsigned integer. Indicates the status of a registration in the NA response. MUST be set to 0 in NS messages. See below.

Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Registration Lifetime: 16-bit unsigned integer. The amount of time in a unit of 10 seconds that the router should retain the Neighbor Cache entry for the sender of the NS that includes this option.

EUI-64: 64 bits. This field is used to uniquely identify the interface of the registered address by including the EUI-64 identifier [EUI64] assigned to it unmodified.

The Status values used in Neighbor Advertisements are:

Status	Description
0	Success
1	Duplicate Address
2	Neighbor Cache Full
3-255	Allocated using Standards Action [RFC2434]

Table 1

4.2. 6LoWPAN Context Option

The optional 6LoWPAN Context Option (6CO) carries prefix information for LoWPAN header compression, and is similar to the Prefix Information Option of [RFC4861]. However, the prefixes can be remote as well as local to the LoWPAN since header compression potentially applies to all IPv6 addresses. This option allows for the dissemination of multiple contexts identified by a Context Identifier (CID) for use as specified in [I-D.ietf-6lowpan-hc]. A context may be a prefix of any length or an address (/128), and up to 16 6LoWPAN Context options may be carried in an Router Advertisement message.

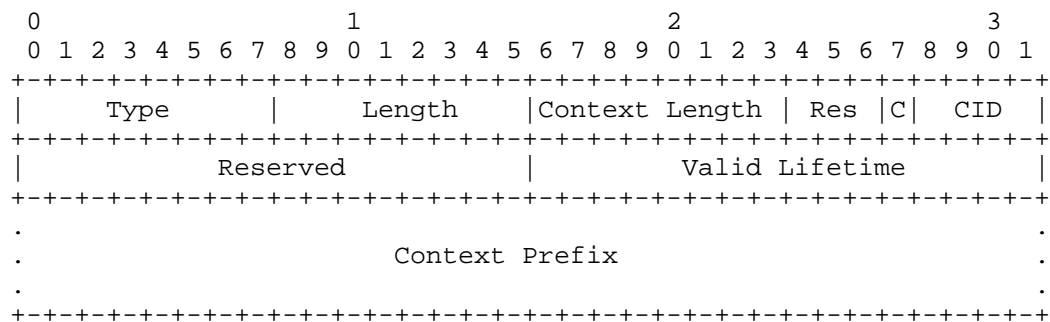


Figure 1: 6LoWPAN Context Option format

Type: TBD2

Length: 8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 bytes. May be 2 or 3 depending on the length of the Context Prefix field.

Context Length: 8-bit unsigned integer. The number of leading bits in the Context Prefix field that are valid. The value ranges from 0 to 128. If it is more than 64 then the Length MUST be 3.

C: 1-bit context compression flag. This flag indicates if the context is valid for use in compression. A context that is not valid MUST NOT be used for compression, but SHOULD be used in decompression in case another compressor has not yet received the updated context information. This flag is used to manage the context lifecycle based on the recommendations in Section 7.2.

CID: 4-bit Context Identifier for this prefix information. CID is used by context based header compression specified in [I-D.ietf-6lowpan-hc]. The list of CIDs for a LoWPAN is configured by on the 6LBR that originates the context information for the LoWPAN.

Res, Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

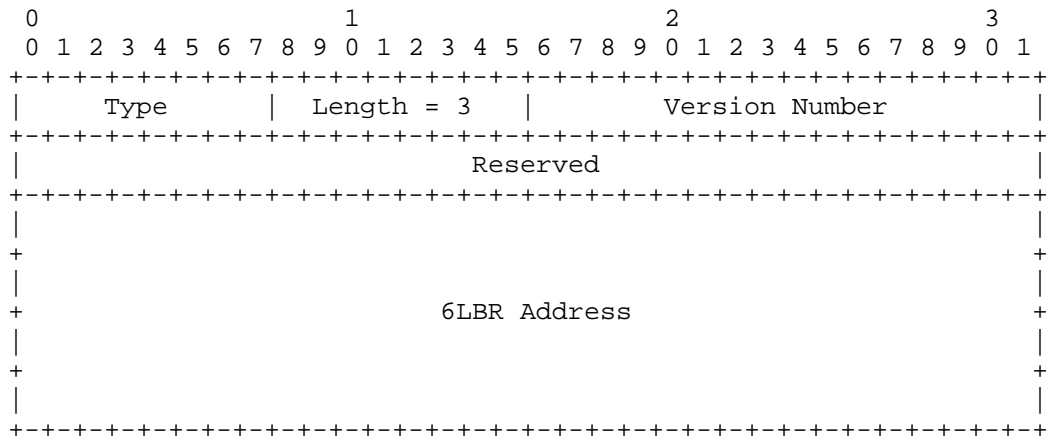
Valid Lifetime: 16-bit unsigned integer. The length of time in a unit of 10 seconds (relative to the time the packet is received) that the context is valid for the purpose of header compression or decompression. A value of all zero bits (0x0) indicates that this context entry MUST be removed immediately.

Context Prefix: The IPv6 prefix or address corresponding to the Context ID (CID) field. The valid length of this field is included in the Context Length field. This field is padded with zeros in order to make the option a multiple of 8-bytes.

4.3. Authoritative Border Router Option

The optional Authoritative Border Router Option (ABRO) is needed when Router Advertisement (RA) messages are used to disseminate prefixes and context information across a route-over topology. In this case 6LRs receive Prefix Information options from other 6LRs. This implies that a 6LR can't just let the most recently received RA win. In order to be able to reliably add and remove prefixes from the LoWPAN we need to carry information from the authoritative 6LBR. This is done by introducing a version number which the 6LBR sets and 6LRs propagate as they propagate the prefix and context information with this Authoritative Border Router Option. When there are multiple 6LBRs they would have separate version number spaces. Thus this option needs to carry the IP address of the 6LBR that originated that set of information.

The Authoritative Border Router option MUST be included in all Router Advertisement messages in the case when Router Advertisements are used to propagate information between routers (as described in Section 8.2).



Fields:

Type: TBD3

Length: 8-bit unsigned integer. The length of the option in units of 8 bytes. Always 3.

Version Number: 16-bit unsigned integer. The version number corresponding to this set of information contained in the RA message. The authoritative 6LBR originating the prefix increases this version number each time its set of prefix or context information changes. This version number uses sequence number arithmetic as it may wrap around.

Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

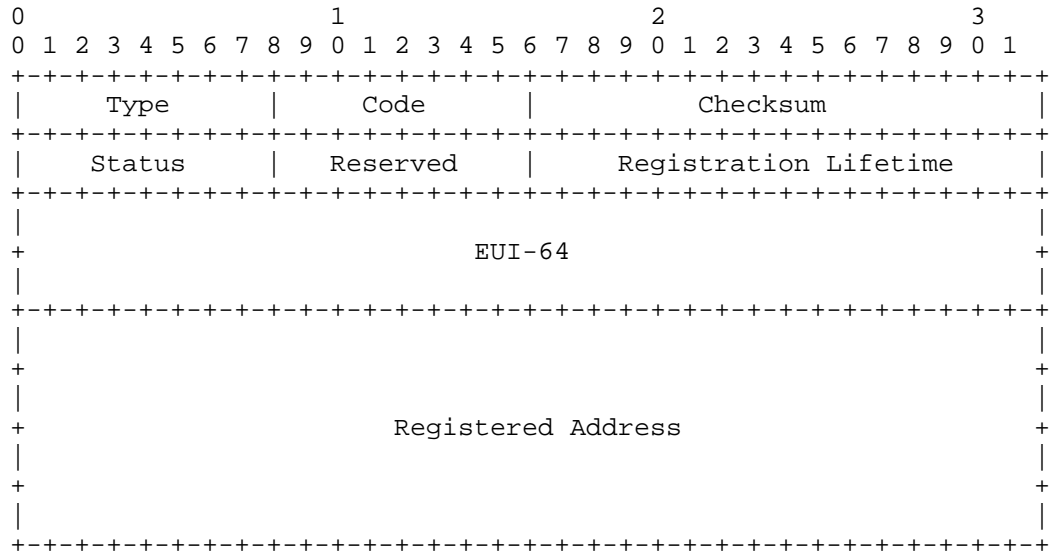
6LBR Address: IPv6 address of the 6LBR that is the origin of the included version number.

4.4. Duplicate Address messages

For the optional multihop DAD exchanges between 6LR and 6LBR specified in Section 8.2 there are two new ICMPv6 message types called the Duplicate Address Request (DAR) and Duplicate Address Confirmation (DAC). We avoid reusing the Neighbor Solicitation and Neighbor Advertisement messages for this purpose since these messages are not subject to the hop limit=255 check as they are forwarded by intermediate 6LRs. The information contained in the messages are otherwise the same as would be in a Neighbor Solicitation carrying a Address Registration option, with the message format inlining the

fields that are in the ARO.

The DAR and DAC use the same message format with different ICMPv6 type values, and the Status field is only meaningful in the DAC message.



IP fields:

IPv6 source: A non link-local address of the sending router.

IPv6 destination: A non link-local address of the sending router.
In a DAC this is just the source from the DAR.

Hop Limit: Set to MULTIHOPI_HOPLIMIT on transmit. MUST be ignored on receipt.

ICMP Fields:

Type: TBD4 for DAR and TBD5 for DAC

Code: Set to zero on transmit. MUST be ignored on receipt.

Checksum: The ICMP checksum. See [RFC4443].

Status: 8-bit unsigned integer. Indicates the status of a registration in the DAC. MUST be set to 0 in DAR. See Table 1.

- Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.
- Registration Lifetime: 16-bit unsigned integer. The amount of time in a unit of 10 seconds that the router should retain the Neighbor Cache entry for the sender of the NS that includes this option. A value of 0 indicates in an NS that the neighbor cache entry should be removed.
- EUI-64: 64 bits. This field is used to uniquely identify the interface of the registered address by including the EUI-64 identifier [EUI64] assigned to it unmodified.
- Registered Address: 128-bit field. Carries the host address, which was contained in the IPv6 Source field in the NS that contained the ARO option sent by the host.

5. Host Behavior

Hosts in a LowPAN use the Address Registration option in the Neighbor Solicitation messages they send as a way to maintain the Neighbor Cache in the routers thereby removing the need for multicast Neighbor Solicitations to do address resolution. Unlike in [RFC4861] the hosts initiate updating the information they receive in Router Advertisements by sending Router Solicitations before the information expires. Finally, when Neighbor Unreachability Detection indicates that one or all default routers have become unreachable, then the host uses Router Solicitations to find a new set of default routers.

5.1. Forbidden Actions

A host in a 6LowPAN MUST NOT accept a Redirect message. Redirect messages are problematic on a link with non-transitive reachability.

A host would never multicast a Neighbor Solicitation message.

5.2. Interface Initialization

When the interface on a host is initialized it follows the specification in [RFC4861]. A link-local address is formed based on the EUI-64 identifier [EUI64] assigned to the interface as per [RFC4944] or the appropriate IP-over-foo document for the link, and then the host sends Router Solicitation messages as described in [RFC4861] Section 6.3.7.

There is no need to join the Solicited-Node multicast address since nobody multicasts Neighbor Solicitations in this type of network. A

host MUST join the all-nodes multicast address.

5.3. Sending a Router Solicitation

The Router Solicitation is formatted as specified in [RFC4861] and sent to the IPv6 All-Routers multicast address (see [RFC4861] Section 6.3.7 for details). An SLLA option MUST be included to enable unicast Router Advertisements in response. An unspecified source address MUST NOT be used in RS messages.

If the link layer supports a way to send packets to some kind of all-routers anycast link-layer address, then that MAY be used to convey these packets to a router.

Since hosts do not depend on multicast Router Advertisements to discover routers, the hosts need to intelligently retransmit Router Solicitations whenever the default router list is empty, one of its default routers becomes unreachable, or the lifetime of the prefixes and contexts in the previous RA are about to expire. The RECOMMENDED retransmissions is to initially send up to 3 (MAX_RTR_SOLICITATIONS) RS messages separated by at least 10 seconds (RTR_SOLICITATION_INTERVAL) as specified in [RFC4861], and then switch to slower retransmissions. After the initial retransmissions the host SHOULD do binary exponential backoff of the retransmission timer for each subsequent retransmission. However, it is useful to have a maximum retransmission timer of 60 seconds (MAX_RTR_SOLICITATION_INTERVAL). In all cases the RS retransmissions are terminated when a RA is received.

5.4. Processing a Router Advertisement

The processing of Router Advertisements is as in [RFC4861] with the addition of handling the 6LoWPAN Context option and triggering address registration when a new address has been configured. Furthermore, the SLLA option MUST be included in the RA. Unlike in [RFC4861], the maximum value of the RA Router Lifetime field MAY be up to 0xFFFF (approximately 18 hours).

Should the host erroneously receive a Prefix Information option with the 'L' (on-link) flag set, then that Prefix Information Option (PIO) MUST be ignored.

5.4.1. Address configuration

Address configuration follows [RFC4862]. For an address not derived from an EUI-64, the M flag of the RA determines how the address can be configured. If the M flag is set in the RA, then DHCPv6 MUST be used to assign the address. If the M flag is not set, then the

address can be configured by any other means (and duplicate detection is performed as part of the registration process).

Once an address has been configured it will be registered by unicasting a Neighbor Solicitation with the Address Registration option to one or more routers.

5.4.2. Storing Contexts

The host maintains a conceptual data structure for the context information it receives from the routers, which is called the Context Table. This includes the Context ID, the prefix (from the Context Prefix field in the 6CO), the Compression bit, and the Valid Lifetime. A Context Table entry that has the Compression bit clear is used for decompression when receiving packets, but MUST NOT be used for compression when sending packets.

When a 6CO option is received in a Router Advertisement it is used to add or update the information in the Context Table. If the Context ID field in the 6CO matches an existing Context Table entry, then that entry is updated with the information in the 6CO. If the Valid Lifetime field in the 6CO is zero, then the entry is immediately deleted.

If there is no matching entry in the Context Table, and the Valid Lifetime field is non-zero, then a new context is added to the Context Table. The 6CO is used to update the created entry.

When the 6LBR changes the context information a host might not immediately notice. And in the worst case a host might have stale context information. For this reason 6LBRs use the recommendations in Section 7.2 for carefully managing the context lifecycle. Nodes should be careful about using header compression in RA messages that include 6COs.

5.4.3. Maintaining Prefix and Context Information

The prefix information is timed out as specified in [RFC4861]. When the Valid Lifetime for a Context Table entry expires the entry is deleted.

A host should inspect the various lifetimes to determine when it should next initiate sending a Router Solicitation to ask for any updates to the information. The lifetimes that matter are the Default Router lifetime, the Valid Lifetime in the Prefix Information options, and the Valid Lifetime in the 6CO. The host SHOULD unicast one or more Router Solicitations to the router well before the minimum of those lifetimes (across all the prefixes and all the

contexts) expire, and switch to multicast RS messages if there is no response to the unicasts. The retransmission behavior for the Router Solicitations is specified in section Section 5.3.

5.5. Registration and Neighbor Unreachability Detection

Hosts send Unicast Neighbor Solicitation (NS) messages to register their IPv6 addresses, and also to do NUD to verify that their default routers are still reachable. The registration is performed by the host including an ARO in the Neighbor Solicitation it sends. Even if the host doesn't have data to send, but is expecting others to try to send packets to the host, the host needs to maintain its Neighbor Cache entries in the routers. This is done by sending NS messages with the ARO to the router well in advance of the registration lifetime expiring. NS messages are retransmitted up to MAX_UNICAST_SOLICIT times using a minimum timeout of RETRANS_TIMER until the host receives an Neighbor Advertisement message with an ARO option.

Hosts that receive Router Advertisement messages from multiple default routers SHOULD attempt to register with more than one of them in order to increase the robustness of the network.

Note that Neighbor Unreachability Detection probes can be suppressed by Reachability Confirmations from transport protocols or applications as specified in [RFC4861].

When a host knows it will no longer use a router it is registered, it SHOULD de-register with the router by sending an NS with an ARO containing a lifetime of 0.

5.5.1. Sending a Neighbor Solicitation

The host triggers sending Neighbor Solicitation (NS) messages containing an ARO when a new address is configured, when it discovers a new default router, or well before the Registration Lifetime expires. Such an NS MUST include a Source Link-Layer Address (SLLA) option, since the router needs to record the link-layer address of the host. An unspecified source address MUST NOT be used in NS messages.

5.5.2. Processing a Neighbor Advertisement

A host handles Neighbor Advertisement messages as specified in [RFC4861], with added logic described in this section for handling the Address Registration option.

In addition to the normal validation of a Neighbor Advertisement and

its options, the Address Registration option is verified as follows (if present). If the Length field is not two, the option is silently ignored. If the EUI-64 field does not match the EUI-64 of the interface, the option is silently ignored.

If the status field is zero, then the address registration was successful. The host saves the Registration Lifetime from the Address Registration option for use to trigger a new NS well before the lifetime expires. If the Status field is not equal to zero, the address registration has failed.

5.5.3. Recovering from Failures

The procedure for maintaining reachability information about a neighbor is the same as in [RFC4861] Section 7.3 with the exception that address resolution is not performed.

The address registration procedure may fail for two reasons: no response to Neighbor Solicitations is received (NUD failure), or an Address Registration option with a failure Status (Status > 0) is received. In the case of NUD failure the entry for that router will be removed thus address registration is no longer of importance. When an Address Registration option with a non-zero Status field is received this indicates that registration for that address has failed. A failure Status of one indicates that a duplicate address was detected and the procedure described in [RFC4862] Section 5.4.5 is followed. The host MUST NOT use the address it tried to register. If the host has valid registrations with other routers, these MUST be removed by registering with each using a zero ARO lifetime.

A Status code of two indicates that the Neighbor Cache of that router is full. In this case the host SHOULD remove this router from its default router list and attempt to register with another router. If the host has no more default routers it needs to revert to sending Router Solicitations as specified in section Section 5.3.

Other failure codes may be defined in future documents.

5.6. Next-hop Determination

The IP address of the next-hop for a destination is determined as follows. Destinations to the link-local prefix (FE80::) are always sent on the link to that destination. It is assumed that link-local addresses are formed as specified in Section 5.2 from the EUI-64, and address resolution is not performed.

All other prefixes are assumed to be off-link [RFC5889]. Anycast addresses are always considered to be off-link. They are therefore

sent to one of the routers in the Default Router List.

Multicast addresses are considered to be on-link and are resolved as specified in [RFC4944] or the appropriate IP-over-foo document.

A LoWPAN Node is not required to maintain a minimum of one buffer per neighbor as specified in [RFC4861], since packets are never queued while waiting for address resolution.

5.7. Address Resolution

The address registration mechanism and the SLLA option in Router Advertisement messages provide sufficient a priori state in routers and hosts to resolve an IPv6 address to its associated link-layer address. As all prefixes but the link-local prefix are always assumed to be off-link, multicast-based address resolution between neighbors is not needed.

Link-layer addresses for neighbors are stored in Neighbor Cache entries [RFC4861]. In order to achieve LoWPAN compression, most global addresses are formed using a link-layer address. Thus a host can minimize memory usage by optimizing for this case and only storing link-layer address information if it differs from the link-layer address corresponding to the Interface ID of the IPv6 address (i.e., differs in more than the on-link/global bit being inverted).

5.8. Sleeping

It is often advantageous for battery-powered hosts in LoWPANs to keep a low duty cycle. The optimizations described in this document enable hosts to sleep as described further in this section. Routers may want to cache traffic destined to a host which is sleeping, but such functionality is out of the scope of this document.

5.8.1. Picking an Appropriate Registration Lifetime

As all Neighbor Discovery messages are initiated by the hosts, this allows a host to sleep or otherwise be unreachable between NS/NA message exchanges. The Address Registration option attached to NS messages indicates to a router to keep the Neighbor Cache entry for that address valid for the period in the Registration Lifetime field. A host should choose a sleep time appropriate for its energy characteristics, and set a registration lifetime larger than the sleep time to ensure the registration is renewed successfully (considering e.g. clock drift and additional time for potential retransmissions of the re-registration). A host should also consider the stability of the network (how quickly the topology changes) when choosing its sleep time (and thus registration lifetime). A dynamic

network requires a shorter sleep time so that routers don't keep invalid neighbor cache entries for nodes longer than necessary.

5.8.2. Behavior on Wakeup

When a host wakes up from a sleep period it SHOULD maintain its current address registrations that will timeout before the next wakeup. This is done by sending Neighbor Solicitation messages with the Address Registration option as described in Section 5.5.1. The host may also need to refresh its prefix and context information by sending a new unicast Router Solicitation (the maximum Router Lifetime is about 18 hours whereas the maximum Registration lifetime is about 7 days). If after wakeup the host (using NUD) determines that some or all previous default routers have become unreachable, then the host will send multicast Router Solicitations to discover new default router(s) and restart the address registration process.

6. Router Behavior for 6LR and 6LBR

Both 6LRs and 6LBRs maintain the Neighbor Cache [RFC4861] based on the Address Registration Options they receive in Neighbor Advertisement messages from hosts, Neighbor Discovery packets from other nodes, and potentially a routing protocol used in the 6LoWPAN as outlined in Section 3.5.

The routers SHOULD NOT garbage collect Registered Neighbor Cache entries (see Section 3.4) since they need to retain them until the Registration Lifetime expires. Similarly, if Neighbor Unreachability Detection on the router determines that the host is UNREACHABLE (based on the logic in [RFC4861]), the Neighbor Cache entry SHOULD NOT be deleted but be retained until the Registration Lifetime expires. A renewed ARO should mark the cache entry as STALE. Thus for 6LoWPAN Routers the Neighbor Cache doesn't behave like a cache. Instead it behaves as a registry of all the host addresses that are attached to the Router.

Routers MAY implement the Default Router Preferences [RFC4191] and use that to indicate to the host whether the router is a 6LBR or a 6LR. If this is implemented then 6LRs with no route to a border router MUST set Prf to (11) for low preference, other 6LRs MUST set Prf to (00) for normal preference, and 6LBRs MUST set Prf to (01) for high preference.

6.1. Forbidden Actions

A router SHOULD NOT send Redirect messages. Since the link has non-transitive reachability the router has no way to determine that the

recipient of a Redirect message can reach the link-layer address.

A router MUST NOT set the 'L' (on-link) flag in the Prefix Information options, since that might trigger hosts to send multicast Neighbor Solicitations.

6.2. Interface Initialization

A router initializes its interface more or less as in [RFC4861]. However, a 6LR might want to wait to make its interfaces advertising (implicitly keeping the AdvSendAdvertisements flag clear) until it has received the prefix(es) and context information from its 6LBR. That is independent of whether prefixes and context information is disseminated using the methods specified in this document, or using some other method.

6.3. Processing a Router Solicitation

A router processes Router Solicitation messages as specified in [RFC4861]. The differences relate to the inclusion of Authoritative Border Router options in the Router Advertisement (RA) messages, and the exclusive use of unicast Router Advertisements. If a 6LR has received an ABRO from a 6LBR, then it will include that option unmodified in the Router Advertisement messages it sends. And if the 6LR has received RAs, whether with the same prefixes and context information or different, from a different 6LBR, then it will need to keep those prefixes and context information separately so that the RAs the 6LR sends will maintain the association between the ABRO and the prefixes and context information. The router can tell which 6LBR originated the prefixes and context information from the 6LBR Address field in the ABRO. When a router has information tied to multiple ABROs, a single RS will result in multiple RAs each containing a different ABRO.

A Router Solicitation might be received from a host that has not yet registered its address with the router. Thus the router MUST NOT modify an existing Neighbor Cache entry based on the SLLA option from the Router Solicitation. However, a router MAY create a Tentative Neighbor Cache entry based on the SLLA option. Such a Tentative Neighbor Cache entry SHOULD be timed out in TENTATIVE_NCE_LIFETIME seconds unless a registration converts it into a Registered NCE.

A 6LR or 6LBR MUST include a Source Link-layer address option in the Router Advertisements it sends. That is required so that the hosts will know the link-layer address of the router. Unlike in [RFC4861], the maximum value of the RA Router Lifetime field MAY be up to 0xFFFF (approximately 18 hours).

Unlike [RFC4861] which suggests multicast Router Advertisements, this specification optimizes the exchange by always unicasting RAs in response to RSs. This is possible since the RS always includes a SLLA option, which is used by the router to unicast the RA.

6.4. Periodic Router Advertisements

A router does not need to send any periodic Router Advertisement messages since the hosts will solicit updated information by sending Router Solicitations before the lifetimes expire.

However, if the routers use Router Advertisements to optionally distribute prefix and/or context information across a route-over topology, that might require periodic Router Advertisement messages. Such RAs are sent using the configurable MinRtrAdvInterval and MaxRtrAdvInterval as per [RFC4861].

6.5. Processing a Neighbor Solicitation

A router handles Neighbor Solicitation messages as specified in [RFC4861], with added logic described in this section for handling the Address Registration option.

In addition to the normal validation of a Neighbor Solicitation and its options, the Address Registration option is verified as follows (if present). If the Length field is not two, or if the Status field is not zero, then the Neighbor Solicitation is silently ignored.

If the source address of the NS is the unspecified address, or if no SLLA option is included, then any included ARO is ignored, that is, the NS is processed as if it did not contain an ARO.

6.5.1. Checking for Duplicates

If the NS contains a valid ARO, then the router inspects its Neighbor Cache on the arriving interface to see if it is a duplicate. If there is no Neighbor Cache entry for the IPv6 source address of the NS, then it isn't a duplicate. If there is such a Neighbor Cache entry and the EUI-64 is the same, then it isn't a duplicate either. Otherwise it is a duplicate address. Note that if multihop DAD (Section 8.2) is used then the checks are slightly different to take into account Tentative Neighbor Cache entries. In the case it is a duplicate address then the router responds with a unicast Neighbor Advertisement (NA) message with the ARO Status field set to one (to indicate the address is a duplicate) as described in Section 6.5.2. In this case there is no modification to the Neighbor Cache.

6.5.2. Returning Address Registration Errors

Address registration errors are not sent back to the source address of the NS due to a possible risk of L2 address collision. Instead the NA is sent to the link-local IPv6 address with the IID part derived from the EUI-64 field of the ARO as per [RFC4944]. In particular, this means that the universal/local bit needs to be inverted. The NA is formatted with a copy of the ARO from the NS, but with the Status field set to indicate the appropriate error.

6.5.3. Updating the Neighbor Cache

If ARO did not result in a duplicate address being detected as above, then if the Registration Lifetime is non-zero the router creates (if it didn't exist) or updates (otherwise) a Neighbor Cache entry for the IPv6 source address of the NS. If the Neighbor Cache is full and a new entry needs to be created, then the router responds with a unicast NA with the ARO Status field set to two (to indicate the router's Neighbor Cache is full) as described in Section 6.5.2.

The Registration Lifetime and the EUI-64 are recorded in the Neighbor Cache entry. A unicast Neighbor Advertisement (NA) is then sent in response to the NS. This NA SHOULD include a copy of the ARO, with the Status field set to zero. A TLLA option is not required in the NA, since the host already knows the router's link-layer address from Router Advertisements.

If the ARO contains a zero Registration Lifetime then any existing Neighbor Cache entry for the IPv6 source address of the NS MUST be deleted, and a NA sent as above.

Should the Registration Lifetime in a Neighbor Cache entry expire, then the router MUST delete the cache entry.

The addition and removal of Registered Neighbor Cache entries would result in notifying the routing protocol.

Note: If the optional multihop DAD (Section 8.2) is used, then the updating of the Neighbor Cache is slightly different due to Tentative NCEs.

6.5.4. Next-hop Determination

In order to deliver a packet destined for a 6LN registered with a router, next-hop determination is slightly different for routers than hosts. The routing table is checked to determine the nexthop IP address. A registered NCE determines if the nexthop IP-address is on-link. It is the responsibility of the routing protocol to

maintain on-link information about its registered neighbors.
Tentative NCEs MUST NOT be used to maintain on-link status.

6.5.5. Address Resolution between Routers

There needs to be a mechanism somewhere for the routers to discover each other's link-layer addresses. If the routing protocol used between the routers provides this, then there is no need for the routers to use the Address Registration option between each other. Otherwise, the routers MAY use the ARO. When routers use ARO to register with each other and the optional multihop DAD Section 8.2 is in use, then care should be taken to ensure that there isn't a flood of ARO-carrying messages sent to the 6LBR as each router hears an ARO from their neighboring routers. The details for this is out of scope of this document.

Optionally Routers can use multicast Neighbor Solicitations as in [RFC4861] to resolve each others link-layer addresses. Thus Routers MAY multicast Neighbor Solicitations for other routers, for example as a result of receiving some routing protocol update. Routers MUST respond to multicast Neighbor Solicitations. This implies that Routers MUST join the Solicited-node multicast addresses as specified in [RFC4861].

7. Border Router Behavior

A 6LBR handles sending of Router Advertisements and processing of Neighbor Solicitations from hosts as specified above in section Section 6. A 6LBR SHOULD always include an Authoritative Border Router option in the Router Advertisements it sends, listing itself as the 6LBR Address. That requires that the 6LBR maintain the version number in stable storage, and increases the version number when some information in its Router Advertisements change. The information whose change affects the version are in the Prefix Information options (the prefixes or their lifetimes) and in the 6CO option (the prefixes, Context IDs, or lifetimes.)

In addition, a 6LBR is somehow configured with the prefix or prefixes that are assigned to the LoWPAN, and advertises those in Router Advertisements as in [RFC4861]. Optionally, in the case of route-over, those prefixes can be disseminated to all the 6LRs using the technique in Section 8.1. However, there might be mechanisms outside of the scope of this document that can be used instead for prefix dissemination with route-over.

If the 6LoWPAN uses Header Compression [I-D.ietf-6lowpan-hc] with context then the 6LBR needs to manage the context IDs, and advertise

those in Router Advertisements by including 6CO options in its Router Advertisements so that directly attached hosts are informed about the context IDs. Below we specify things to consider when the 6LBR needs to add, remove, or change the context information. Optionally, in the case of route-over, the context information can be disseminated to all the 6LRs using the technique in Section 8. However, there might be mechanisms outside of the scope of this document that can be used instead for disseminating context information with route-over.

7.1. Prefix Determination

The prefix or prefixes used in a LoWPAN can be manually configured, or can be acquired using DHCPv6 Prefix Delegation [RFC3633]. For a LoWPAN that is isolated from the network, either permanently or occasionally, the 6LBR can assign a ULA prefix using [RFC4193]. The ULA prefix should be stored in stable storage so that the same prefix is used after a failure of the 6LBR. If the LoWPAN has multiple 6LBRs, then they should be configured with the same set of prefixes. The set of prefixes are included in the Router Advertisement messages as specified in [RFC4861].

7.2. Context Configuration and Management

If the LoWPAN uses Header Compression [I-D.ietf-6lowpan-hc] with context then the 6LBR may be configured with context information and related context IDs. If the LoWPAN has multiple 6LBRs, then they MUST be configured with the same context information and context IDs.

The context information carried in Router Advertisement (RA) messages originate at 6LBRs and must be disseminated to all the routers and hosts within the LoWPAN. RAs include one 6CO for each context.

For the dissemination of context information using the 6CO, a strict lifecycle SHOULD be used in order to ensure the context information stays synchronized throughout the LoWPAN. New context information SHOULD be introduced into the LoWPAN with C=0, to ensure it is known by all nodes that may have to decompress based on this context information. Only when it is reasonable to assume that this information was successfully disseminated SHOULD an option with C=1 be sent, enabling the actual use of the context information for compression.

Conversely, to avoid that nodes send packets making use of previous values of contexts, resulting in ambiguity when receiving a packet that uses a recently changed context, old values of a context SHOULD be taken out of use for a while before new values are assigned to this specific context. That is, in preparation for a change of context information, its dissemination SHOULD continue for at least

MIN_CONTEXT_CHANGE_DELAY with C=0. Only when it is reasonable to assume that the fact that the context is now invalid was successfully disseminated, should the context ID be taken out of dissemination or reused with a different Context Prefix field. In the latter case, dissemination of the new value again SHOULD start with C=0, as above.

8. Optional Behavior

Optionally the Router Advertisement messages can be used to disseminate prefixes and context information to all the 6LRs in a route-over topology. If all routers are configured to use another mechanism for such information distribution, this mechanism MAY stay unused.

There is also the option for a 6LR to perform multi-hop DAD (for non-EUI-64 derived IPv6 addresses) against a 6LBR in a route-over topology by using the DAR and DAC messages. This is optional because there might be other ways to either allocate unique address, such as DHCPv6 [RFC3315], or other future mechanisms for multihop DAD.

8.1. Multihop Prefix and Context Distribution

The multihop distribution relies on Router Solicitation messages and Router Advertisement (RA) messages sent between routers, and using the ABRO version number to control the propagation of the information (prefixes and context information) that is being sent in the RAs.

This multihop distribution mechanism can handle arbitrary information from an arbitrary number of 6LBRs. However, the semantics of the context information requires that all the 6LNs use the same information, whether they send, forward, or receive compressed packets. Thus the manager of the 6LBRs need to somehow ensure that the context information is in synchrony across the 6LBRs. This can be handled in different ways. One possible way to ensure it is to treat the context and prefix information as originating from some logical or virtual source, which in essence means that it looks like the information is distributed from a single source.

If a set of 6LBRs behave as a single one (using mechanisms out of scope of this document) so that the prefixes and contexts and ABRO version number will be the same from all the 6LBRs, then those 6LBRs can pick a single IP address to use in the ABRO option.

8.1.1. 6LBRs Sending Router Advertisements

6LBRs supporting multihop prefix and context distribution MUST include an ABRO in each of its RAs. The ABRO Version Number field is

used to keep prefix and context information consistent throughout the LoWPAN along with the guidelines in Section 7.2. Each time any information in the set of PIO or 6CO options change, the ABRO Version is increased by one.

This requires that the 6LBR maintain the PIO, 6CO, and ABRO Version Number in stable storage, since an old version number will be silently ignored by the 6LRs.

8.1.2. Routers Sending Router Solicitations

If multihop distribution is done using Router Advertisement (RA) messages, then on interface initialization a router SHOULD send some Router Solicitation messages similarly to how hosts do this in [RFC4861]. That will cause the routers to respond with RA messages which then can be used to initially seed the prefix and context information.

8.1.3. Routers Processing Router Advertisements

If multihop distribution is not done using RA messages, then the routers follow [RFC4861] which states that they merely do some consistency checks and nothing in Section 8.1 applies. Otherwise the routers will check and record the prefix and context information from the received RAs, and use that information as follows.

If a received RA does not contain a Authoritative Border Router option, then the RA MUST be silently ignored.

The router uses the 6LBR Address field in the ABRO to check if it has previously received information from the 6LBR. If it finds no such information, then it just records the 6LBR Address and Version and the associated prefixes and context information. If the 6LBR is previously known, then the Version number field MUST be compared against the recorded version number for that 6LBR. The comparison MUST be done the same way as TCP sequence number comparisons to handle the case when the version number wraps around. If the version number received in the packet is less than the stored version number (following [RFC1982] Section 3.2), then the information in the RA is silently ignored. Otherwise the recorded information and version number are updated.

By TCP sequence number comparison we mean that half of the version number space is "old" and half is "new". For example, if the current version number is 0x2, then anything between 0x80000003 (0x2-0x7fffffff) and 0x1 is old, and anything between 0x3 and 0x80000002 (0x2+0x80000000) is new.

8.1.4. Storing the Information

The router keeps state for each 6LBR that it sees with an ABRO. This includes the version number, and the complete set of Prefix Information options and 6LoWPAN Context options. The prefixes are timed out based on the Valid lifetime in the Prefix Information Option. The Context Prefix is timed out based on the Valid lifetime in the 6LoWPAN Context option.

While the prefixes and context information are stored in the router their valid and preferred lifetimes are decremented as time passes. This ensures that when the router is in turn later advertising that information in the Router Advertisements it sends, the 'expiry time' doesn't accidentally move further into the future. For example, if a 6CO with a Valid lifetime of 10 minutes is received at time T, and the router includes this in a RA it sends at time T+5 minutes, the Valid lifetime in the 6CO it sends will be only 5 minutes.

8.1.5. Sending Router Advertisements

If multihop distribution is performed using RA messages, then the routers MUST ensure that the ABRO always stay together with the prefixes and context information received with that ABRO. Thus if the router has received prefix P1 with ABRO saying it is from one 6LBR, and prefix P2 from another 6LBR, then the router MUST NOT include the two prefixes in the same RA message. Prefix P1 MUST be in a RA that include a ABRO from the first 6LBR etc. Note that multiple 6LBRs might advertise the same prefix and context information, but they still need to be associated with the 6LBRs that advertised them.

The routers periodically send Router Advertisements as in [RFC4861]. This is for the benefit of the other routers receiving the prefixes and context information. And the routers also respond to Router Solicitations by unicasting RA messages. In both cases the above constraint of keeping the ABRO together with 'its' prefixes and context information apply.

When a router receives new information from a 6LBR, that is, either it hears from a new 6LBR (a new 6LBR Address in the ABRO) or the ABRO version number of an existing 6LBR has increased, then it is useful to send out a few triggered updates. The recommendation is to behave the same as when an interface has become an advertising interface in [RFC4861], that is, send up to three RA messages. This ensures rapid propagation of new information to all the 6LRs.

8.2. Multihop Duplicate Address Detection

The ARO can be used, in addition to registering an address in a 6LR, to have the 6LR verify that the address isn't used by some other host known to the 6LR. However, that isn't sufficient in a route-over topology (or in a LoWPAN with multiple 6LBRs) since some host attached to another 6LR could be using the same address. There might be different ways for the 6LRs to coordinate such Duplicate Address Detection in the future, or addresses could be assigned using a DHCPv6 server that verifies uniqueness as part of the assignment.

This specification offers an optional and simple technique for 6LRs and 6LBRs to perform Duplicate Address Detection that reuses the information from Address Registration option in the DAR and DAC messages. This technique is not needed when the Interface ID in the address is based on an EUI-64, since those are assumed to be globally unique. The technique assumes that the 6LRs either register with all the 6LBRs, or that the network uses some out-of-scope mechanism to keep the DAD tables in the 6LBRs synchronized.

The multihop DAD mechanism is used synchronously the first time an address is registered with a particular 6LR. That is, the ARO option is not returned to the host until multihop DAD has been completed against the 6LBRs. For existing registrations in the 6LR the multihop DAD needs to be repeated against the 6LBRs to ensure that the entry for the address in the 6LBRs does not time out, but that can be done asynchronously with the response to the hosts. For instance, by tracking how much is left of the lifetime the 6LR registered with the 6LBRs and re-registering with the 6LBR when this lifetime is about to run out.

For the synchronous multihop DAD the 6LR performs some additional checks to ensure that it has a Neighbor Cache entry it can use to respond to the host when it receives a response from a 6LBR. This consists of checking for an already existing (Tentative or Registered) Neighbor Cache entry for the registered address with a different EUI-64. If such a Registered NCE exists, then the 6LR SHOULD respond that the address is a duplicate. If such a Tentative NCE exists, then the 6LR SHOULD silently ignore the ARO thereby relying on the host retransmitting the ARO. (This is needed to handle the case when multiple hosts try to register the same IPv6 address at the same time.) If no Neighbor Cache entry exists, then the 6LR MUST create a Tentative Neighbor Cache entry with the EUI-64 and the SLLAO. This entry will be used to send the response to the host when the 6LBR responds positively.

When a 6LR receives a Neighbor Solicitation containing an Address Registration option with a non-zero Registration Lifetime and it has

no existing Registered Neighbor Cache entry, then with this mechanism the 6LR will invoke synchronous multihop DAD.

The 6LR will unicast a Duplicate Address Request message to one or more 6LBRs, where the DAR contains the host's address in the Registered Address field. The DAR will be forwarded by 6LRs until it reaches the 6LBR, hence its IPv6 hop limit field will not be 255 when received by the 6LBR. The 6LBR will respond with a Duplicate Address Confirmation message, which will have a hop limit less than 255 when it reaches the 6LR.

When the 6LR receives the DAC from the 6LBR, it will look for a matching (same IP address and EUI-64) (Tentative or Registered) Neighbor Cache entry. If no such entry is found then the DAC is silently ignored. If an entry is found and the DAC had Status=0 then the 6LR will mark the Tentative Neighbor Cache entry as Registered. In all cases when an entry is found then the 6LR will respond to the host with an NA, copying the Status and EUI-64 fields from the DAC to an ARO option in the NA. In case the status is an error, then the destination IP address of the NA is derived from the EUI-64 field of the DAC.

A Tentative Neighbor Cache entry SHOULD be timed out `TENTATIVE_NCE_LIFETIME` seconds after it was created in order to allow for another host to attempt to register the IPv6 address.

8.2.1. Message Validation for DAR and DAC

A node MUST silently discard any received Duplicate Address Request and Confirmation messages that do not satisfy all of the following validity checks:

- o If the message includes an IP Authentication Header, the message authenticates correctly.
- o ICMP Checksum is valid.
- o ICMP Code is 0.
- o ICMP length (derived from the IP length) is 32 or more bytes.
- o The Registered Address is not a multicast address.
- o All included options have a length that is greater than zero.
- o The IP source address is not the unspecified address, nor a multicast address.

The contents of the Reserved field, and of any unrecognized options, MUST be ignored. Future, backward-compatible changes to the protocol may specify the contents of the Reserved field or add new options; backward-incompatible changes may use different Code values.

Note that due to the forwarding of the DAR and DAC messages between the 6LR and 6LBR there is no hop limit check on receipt for these ICMPv6 message types.

8.2.2. Conceptual Data Structures

A 6LBR implementing the optional multihop DAD needs to maintain some state separate from the Neighbor Cache. We call this conceptual data structure the DAD table. It is indexed by the IPv6 address - the Registered Address in the DAR - and contains the EUI-64 and the registration lifetime of the host that is using that address.

8.2.3. 6LR Sending a Duplicate Address Request

When a 6LR that implements the optional multihop DAD receives an NS from a host and subject to the above checks, the 6LR forms and sends a DAR to at least one 6LBR. The DAR contains the following information:

- o In the IPv6 source address, a global address of the 6LR.
- o In the IPv6 destination address, the address of the 6LBR.
- o In the IPv6 hop limit, MULTIHOP_HOPLIMIT.
- o The Status field MUST be set to zero
- o The EUI-64 and Registration lifetime are copied from the ARO received from the host.
- o The Registered Address set to the IPv6 address of the host, that is, the sender of the triggering NS.

When a 6LR receives an NS from a host with a zero Registration Lifetime then, in addition to removing the Neighbor Cache entry for the host as specified in section Section 6, an DAR is sent to the 6LBRs as above.

A router MUST NOT modify the Neighbor Cache as a result of receiving a Duplicate Address Request.

8.2.4. 6LBR Receiving a Duplicate Address Request

When a 6LBR that implements the optional multihop DAD receives an DAR from a 6LR, it performs the message validation specified in Section 8.2.1. If the DAR is valid the 6LBR proceeds to look for the Registration Address in the DAD Table. If an entry is found and the recorded EUI-64 is different than the EUI-64 in the DAR, then it returns a DAC NA with the Status set to 1 ('Duplicate Address'). Otherwise it returns a DAC with Status set to zero and updates the lifetime.

If no entry is found in the DAD Table and the Registration Lifetime is non-zero, then an entry is created and the EUI-64 and Registered Address from the DAR are stored in that entry.

If an entry is found in the DAD Table, the EUI-64 matches, and the Registration Lifetime is zero then the entry is deleted from the table.

In both of the above cases the 6LBR forms an DAC with the information copied from the DAR and the Status field is set to zero. The DAC is sent back to the 6LR i.e., back to the source of the DAR. The IPv6 hop limit is set to MULTI_HOP_LIMIT

8.2.5. Processing a Duplicate Address Confirmation

When a 6LR that implements the optional multihop DAD receives a DAC message, then it first validates the message per Section 8.2.1. For a valid DAC, if there is no Tentative Neighbor Cache entry matching the Registered address and EUI-64, then the DAC is silently ignored. Otherwise, the information in the DAC and in the Tentative Neighbor Cache entry is used to form an NA to send to the host. The Status code is copied from the DAC to the ARO that is sent to the host. In case of the DAC indicates an error (the Status is non-zero), the NA is returned to the host as described in Section 6.5.2 and the Tentative Neighbor Cache entry for the Registered Address is removed. Otherwise it is made into a Registered Neighbor Cache entry.

A router MUST NOT modify the Neighbor Cache as a result of receiving a Duplicate Address Confirmation, unless there is a Tentative Neighbor Cache entry matching the IPv6 address and EUI-64.

8.2.6. Recovering from Failures

If there is no response from a 6LBR after RETRANS_TIMER [RFC4861] then the 6LR would retransmit the DAR to the 6LBR up to MAX_UNICAST_SOLICIT [RFC4861] times. After this the 6LR SHOULD respond to the host with an ARO Status of zero.

9. Protocol Constants

This section defines the relevant protocol constants used in this document based on a subset of [RFC4861] constants. (*) indicates constants modified from [RFC4861] and (+) indicates new constants.

Additional protocol constants are defined in Section Section 4.

6LBR Constants:

MIN_CONTEXT_CHANGE_DELAY+	300 seconds
---------------------------	-------------

6LR Constants:

MAX_RTR_ADVERTISEMENTS	3 transmissions
------------------------	-----------------

MIN_DELAY_BETWEEN_RAS*	10 seconds
------------------------	------------

MAX_RA_DELAY_TIME*	2 seconds
--------------------	-----------

TENTATIVE_NCE_LIFETIME+	20 seconds
-------------------------	------------

Router Constants:

MULTIHOP_HOPLIMIT+	64
--------------------	----

Host Constants:

RTR_SOLICITATION_INTERVAL*	10 seconds
----------------------------	------------

MAX_RTR_SOLICITATIONS	3 transmissions
-----------------------	-----------------

MAX_RTR_SOLICITATION_INTERVAL+	60 seconds
--------------------------------	------------

10. Examples

10.1. Message Examples

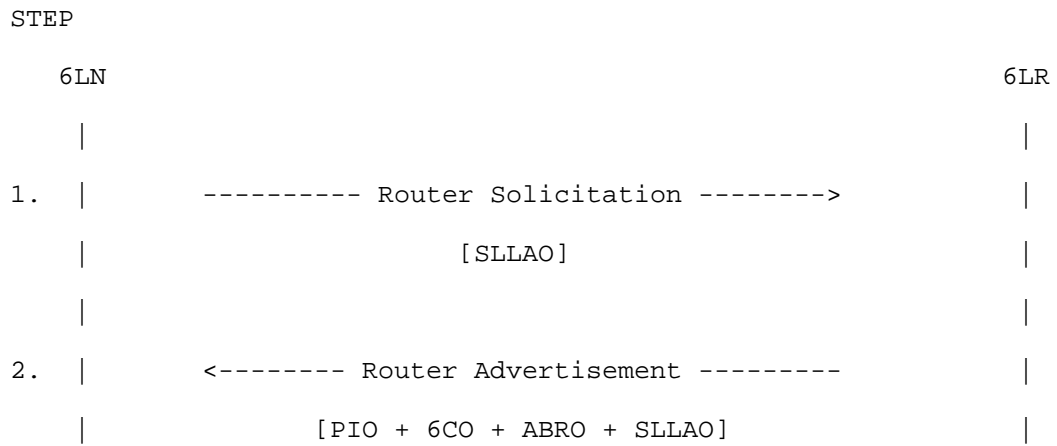


Figure 2: Basic Router Solicitation/Router Advertisement exchange between a node and 6LR or 6LBR

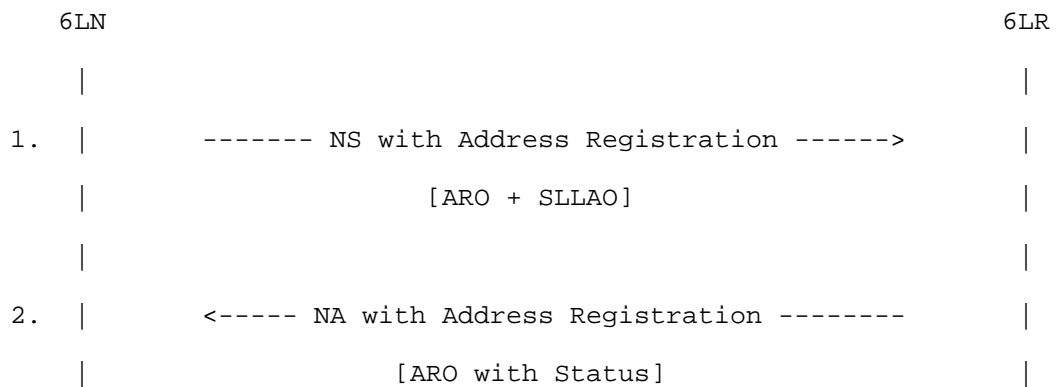


Figure 3: Neighbor Discovery Address Registration

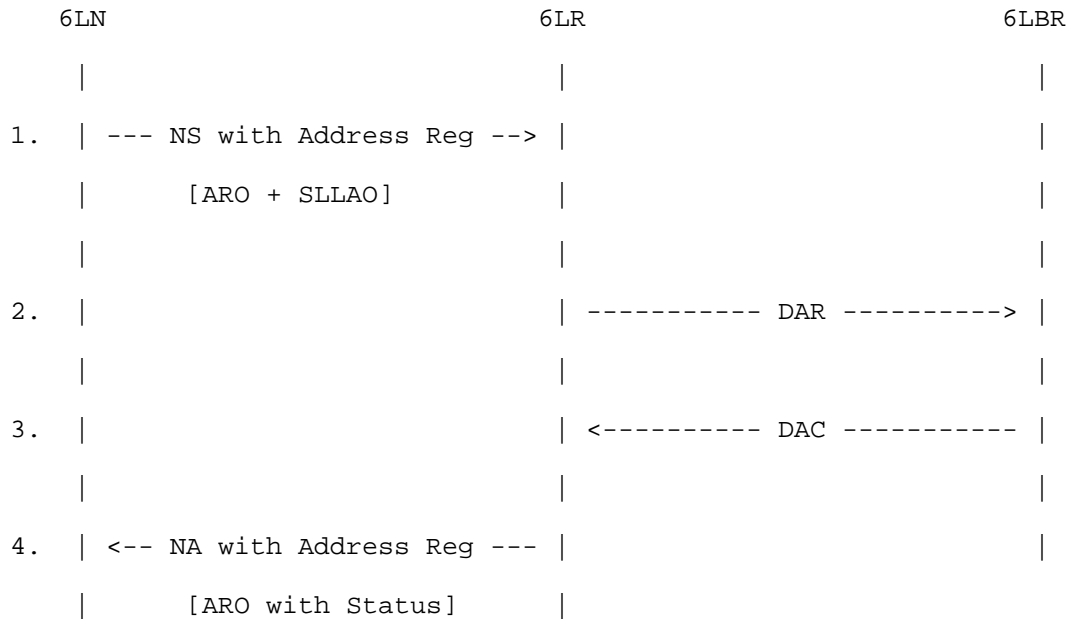


Figure 4: Neighbor Discovery Address Registration with Multi-Hop DAD

10.2. Host Bootstrapping Example

The following example describes the address bootstrapping scenarios using the optimized ND mechanisms specified in this document. It is assumed that the 6LN first performs a sequence of operations in order to get secure access at the link-layer of the LoWPAN and obtain a key for link-layer security. The methods of how to establish the link-layer security is out of scope of this document. In this example an IEEE 802.15.4 6LN forms a 16-bit short-address based IPv6 addresses without using DHCPv6 (i.e., the M flag is not set in the Router Advertisements).

1. After obtaining link-level security, a 6LN assigns a link-local IPv6 address to itself. A link-local IPv6 address is configured based on the 6LN's EUI-64 link-layer address formed as per [RFC4944].

2. Next the 6LN determines one or more default routers in the network by sending an RS to the all-routers multicast address with the SLLA Option set to its EUI-64 link-local address. If the 6LN was able to obtain the link-layer address of a router through its link-layer operations then the 6LN may form a link-local destination IPv6 address for the router and send it a unicast RS. The 6LR responds with a unicast RA to the IP source using the SLLAO from the RS (it

may have created a tentative NCE). See Figure 2.

3. In order to communicate more than one IP hop away the 6LN configures a global IPv6 address. In order to save overhead, this 6LN wishes to configure its IPv6 address based on a 16-bit short address as per [RFC4944]. As the network is unmanaged (M flag not set in RA), the 6LN randomly chooses a 16-bit link-layer address and forms a tentative IPv6 address from it.

4. Next the 6LN registers that address with one or more of its default routers by sending a unicast NS message with an ARO containing its tentative global IPv6 address to register, the registration lifetime and its EUI-64. An SLLAO is also included with the link-layer address corresponding to the address being registered. If a successful (status 0) NA message is received the address can then be used and the 6LN assumes it has been successfully checked for duplicates. If a duplicate address (status 1) NA message is received, the 6LN then removes the temporary IPv6 address and 16-bit link-layer address and goes back to step 3. If a neighbor cache full (status 2) message is received, the 6LN attempts to register with another default router, or if none, goes back to step 2. See Figure 3. Note that an NA message returning an error would be sent back to the link-local EUI-64 based IPv6 address of the 6LN instead of the 16-bit (duplicate) address.

5. The 6LN now performs maintenance by sending a new NS address registration before the lifetime expires.

If multihop DAD and multihop prefix and context distribution is used, the effect of the 6LRs and hosts following the above bootstrapping is a "wavefront" of 6LRs and host being configured spreading from the 6LBRs. First the hosts and 6LRs that can directly reach a 6LBR would receive one or more RAs and configure and register their IPv6 addresses. Once that is done they would enable the routing protocol and start sending out Router Advertisements. That would result in a new set of 6LRs and hosts to receive responses to their Router Solicitations, form and register their addresses, etc. That repeats until all of the 6LRs and hosts have been configured.

10.2.1. Host Bootstrapping Messages

This section brings specific message examples to the previous bootstrapping process. When discussing messages, the following notation is used:

LL64: Link-Local Address based on the EUI-64, which is also the 802.15.4 Long Address.

GP16: Global Address based on the 802.15.4 Short Address. This address may not be unique.

GP64: Global addresses derived from the EUI-64 address as specified in RFC 4944.

MAC64: EUI-64 address used as the link-layer address.

MAC16: IEEE 802.15.4 16-bit short address.

Note that some implementations may use LL64 and GP16 style addresses instead of LL64 and GP64. In the following, we will show an example message flow as to how a node uses LL64 to register a GP16 address for multihop DAD verification.

```

6LN-----RS----->6LR
Src= LL64 (6LN)
Dst= All-router-link-scope-multicast
SLLAO= MAC64 (6LN)

```

```

6LR-----RA----->6LN
Src= LL64 (6LR)
Dst= LL64 (6LN)

```

Note: Source address of RA must be a link-local address (Section 4.2, RFC 4861).

```

6LN-----NS Reg----->6LR
Src= GP16 (6LN)
Dst= LL64 (6LR)
ARO
SLLAO= MAC16 (6LN)

```

```

6LR-----DAR----->6LBR
Src= GP64 or GP16 (6LR)
Dst= GP64 or GP16 (6LBR)
Registered Address= GP16 (6LN) and EUI-64 (6LN)

```

```

6LBR-----DAC----->6LR
Src= GP64 or GP16 (6LBR)
Dst= GP64 or GP16 (6LR)
Copy of information from DAR

```

If Status is a Success:

```

6LR -----NA-Reg----->6LN
Src= LL64 (6LR)
Dst= GP16 (6LN)
ARO with Status = 0

```

If Status is not a success:

```

6LR -----NA-Reg----->6LN
Src= LL64 (6LR)
Dst= LL64 (6LN) --> Derived from the EUI-64 of ARO
ARO with Status > 0

```

Figure 5: Detailed Message Address Examples

10.3. Router Interaction Example

In the Route-over topology, when a routing protocol is run across 6LRs the bootstrapping and neighbor cache management are handled a little differently. The description in this paragraph provides only a guideline for an implementation.

At the initialization of a 6LR, it may choose to bootstrap as a host with the help of a parent 6LR if the optional multihop DAD is performed with the 6LBR. The neighbor cache management of a router and address resolution among the neighboring routers are described in Section 6.5.3 and Section 6.5.5, respectively. In this example, we assume that the neighboring 6lowpan link is secure.

10.3.1. Bootstrapping a Router

In this scenario, the bootstrapping 6LR, 'R1', is multiple hops away from the 6LBR and surrounded by other 6LR neighbors. Initially R1 behaves as a host. It sends multicast RS and receives an RA from one or more neighboring 6LRs. R1 picks one 6LR as its temporary default router and performs address resolution via this default router. Note, if multihop DAD is not required (e.g. in a managed network or using EUI-64 based addresses) then it does not need to pick a temporary default router, however it may still want to send the initial RS message if it wants to autoconfigure its address with the global prefix disseminated by the 6LBR.

Based on the information received in the RAs, R1 updates its cache with entries for all the neighboring 6LRs. Upon completion of the address registration, the bootstrapping router deletes the temporary entry of the default router and the routing protocol is started.

Also note that R1 may refresh its multihop DAD registration directly with the 6LBR (using the nexthop neighboring 6LR determined by the routing protocol for reaching the 6LBR).

10.3.2. Updating the Neighbor Cache

In this example, there are three 6LRs, R1, R2, R3. Initially when R2 boots it sees only R1, and accordingly R2 creates a neighbor cache entry for R1. Now assume R2 receives a valid routing update from router R3. R2 does not have any neighbor cache entry for R3. If the implementation of R2 supports detecting link-layer address from the routing information packets then it directly updates the its neighbor cache using that link-layer information. If this is not possible, then R2 should perform multicast NS with source set with its link-local or global address depending on the scope of the source IP-address received in the routing update packet. The target address of

the NS message is the source IPv6 address of the received routing update packet. The format of the NS message is as described in Section 4.3 of [RFC4861].

More generally any 6LR that receives a valid route-update from a neighboring router for which it does not have any neighbor cache entry is required to update its neighbor cache as described above.

The router (6LR and 6LBR) IP-addresses learned via Neighbor Discovery are not redistributed to the routing protocol.

11. Security Considerations

The security considerations of IPv6 Neighbor Discovery [RFC4861] apply. Additional considerations can be found in [RFC3756].

This specification expects that the link layer is sufficiently protected, for instance using MAC sublayer cryptography. In other words, model 1 from [RFC3756] applies. In particular, it is expected that the LoWPAN MAC provides secure unicast to/from Routers and secure broadcast from the Routers in a way that prevents tampering with or replaying the Router Advertisement messages. However, any future 6LoWPAN security protocol that applies to Neighbor Discovery for 6LoWPAN protocol, is out of scope of this document.

The multihop DAD mechanisms rely on DAR and DAC messages that are forwarded by 6LRs, and as a result the hop_limit=255 check on the receiver does not apply to those messages. This implies that any node on the Internet could successfully send such messages. We avoid any additional security issues due to this by requiring that the routers never modify the Neighbor Cache entry due to such messages, and that they reject them unless they are received on an interface that has been explicitly configured to use these optimizations.

In some future deployments one might want to use SEcure Neighbor Discovery [RFC3971] [RFC3972]. This is possible with the Address Registration option as sent between hosts and routers, since the address that is being registered is the IPv6 source address of the Neighbor Solicitation and SeND verifies the IPv6 source address of the packet. Applying SeND to the optional router-to-router communication in this document is out of scope.

12. IANA Considerations

The document requires three new Neighbor Discovery option types under the subregistry "IPv6 Neighbor Discovery Option Formats":

- o Address Registration Option (TBD1)
- o 6LoWPAN Context Option (TBD2)
- o Authoritative Border Router Option (TBD3)

The document requires two new ICMPv6 types under the subregistry "ICMPv6 type Numbers":

- o Duplicate Address Request (TBD4)
- o Duplicate Address Confirmation (TBD5)

For the purpose of protocol interoperability testing of this specification, the following values are being used temporarily:

- o TBD1 = 31
- o TBD2 = 32
- o TBD3 = 33
- o TBD4 = 155 XXX
- o TBD3 = 156 XXX

This document also requests IANA to create a new registry for the Status values of the Address Registration Option.

[TO BE REMOVED: This registration should take place at the following location: <http://www.iana.org/assignments/icmpv6-parameters>]

13. Guideline for New Features

This section discusses a guideline of new features for implementation and deployment.

Section	Description	Deployment	Implementation
3.1	Host initiated RA	MUST	MUST
3.2	EUI-64 based IPv6-address	MUST	MUST
	16bit-MAC based address	MAY	SHOULD
	Other non-unique addresses	MAY	MAY
3.3	Host Initiated RS	MUST	MUST
	ABRO Processing	SHOULD	MUST
4.1	Registration with ARO	MUST	MUST
4.2, 5.4	6lowpan Context Option	SHOULD	SHOULD
5.1	Re-direct Message Acceptance	MUST NOT	MUST NOT
	Joining Solicited Node		
	Multicast	N/A	N/A
	Joining all-node Multicat	MUST	MUST
?	Using link-layer indication		
	For NUD	SHOULD	MAY
5.5	6lowpan-ND NUD	MUST	MUST
5.8.2	Behavior on wake-up	SHOULD	SHOULD

Figure 6: Guideline for lowpan-nd new-features for hosts

Section	Description	Deployment	Implementation
3.1	Periodic RA	SHOULD NOT	SHOULD NOT
3.2	Address assignment during Startup	SHOULD	MUST
3.3	Supporting EUI-64 based MAC Hosts Supporting 16-bit MAC hosts	MUST MAY	MUST SHOULD
3.4, 4.3 8.1.3, 8.1.4 8.1	ABRO Processing/sending Multihop Prefix storing and re-distribution	MAY	SHOULD
3.5	Tentative NCE	MUST	MUST
8.2	Multihop DAD	MAY	SHOULD
4.1, 6.5 6.5.1 - 6.5.5	ARO Support	MUST	MUST
4.2	6lowpan Context Option	SHOULD	SHOULD
6.3	Process RS/ARO	MUST	MUST

Figure 7: Guideline for 6LR in lowpan-nd

Section	Description	Deployment	Implementation
3.1	Periodic RA	SHOULD NOT	SHOULD NOT
3.2	Address autoconf on Router interface	MUST NOT	MUST NOT
3.3	EUI-64 MAC address support On 6lowpan interface	MUST	MUST
8.1 - 8.1.1			
8.1.5	Multihop Prefix distribution	MAY	SHOULD
8.2	Multihop DAD	MAY	SHOULD

Figure 8: Guideline for 6LBR features

14. Acknowledgments

The authors thank Pascal Thubert, Jonathan Hui, Carsten Bormann, Richard Kelsey, Geoff Mulligan, Julien Abeille, Alexandru Petrescu, Peter Siklosi, Pieter De Mil, Fred Baker, Anthony Schoofs, Phil Roberts, Daniel Gavelle, Joseph Reddy, Robert Cragie, Mathilde Durvy, Colin O'Flynn, Dario Tedeschi and Joakim Eriksson for useful discussions and comments that have helped shaped and improve this document.

Additionally, the authors would like to recognize Carsten Bormann for the suggestions on the Context Prefix Option and contribution to earlier version of the draft, Pascal Thubert for contribution of the original registration idea and extensive contributions to earlier versions of the draft, Jonathan Hui for original ideas on prefix/context distribution and extensive contributions to earlier versions of the draft, Colin O'Flynn for useful Error-to suggestions and contributions to the Examples section, Geoff Mulligan for suggesting the use of Address Registration as part of existing IPv6 Neighbor Discovery messages, and Mathilde Durvy for helping to clarify router interaction.

15. Changelog

Changes from -13 to -14:

- o Introduced the new DAR and DAC ICMPv6 message types for multihop DAD to avoid relying on the Length=4 checks for the ARO. This simplifies implementing the hop limit check.
- o Clarified the hop limit values for the multihop DAD messages by introducing the MULTI_HOP_HOPLIMIT constant set to 64.
- o Clarified when a host should de-register from a router.
- o Added a section on next-hop determination for routers.
- o Removed the infinite lifetime from 6CO.
- o Increased MIN_CONTEXT_CHANGE_DELAY to 300 seconds.

Changes from -12 to -13:

- o Error-to solution added for returning NA messages carrying an error ARO option to the link-local EUI-64 based IPv6 address of the host (#126).
- o New examples added.

Changes from -11 to -12:

- o Version field of ABRO moved after Length for 32-bit alignment of the reserved space (#90).
- o Several clarifications were made on router interaction, including a new section with router interaction examples (#91).
- o Temporary Neighbor Cache Entry created upon host sending NS+ARO, and SLLAO removed from multihop DAD NS/NA messages (#87).

Changes from -10 to -11:

- o Reference to RFC1982 for version number comparison (#80)
- o RA Router Lifetime field use clarified (#81)
- o Make fields 16-bit rather than 32-bit where possible (#83)
- o Unicast RA clarification (#84)
- o Temporary ND option types (#85)
- o SLLA/TLLA clarification (#86)

- o GP16 as source address in initial NS clarification (#87)

Changes from -09 to -10:

- o Clarifications made to Section 8.2 (#66)
- o Explained behavior of Neighbor Cache (#67)
- o Clarified use of SLLAO in RS and NS messages (#68)
- o Added new term 6LN (#69)
- o Small clarification on 6CO flag (#70)
- o Defined host behavior on ARO failure better (#72)
- o Added bootstrapping example for a host (#73)
- o Added new Neighbor Cache Full ARO error (#74)
- o Added rule on the use of the M flag (#75)

Changes from -08 to -09:

- o Clean re-write of the draft (re-use of some introductory material)
- o Merged in draft-chakrabarti-6lowpan-ipv6-nd-simple-00
- o Changed address registration to an option piggybacked on NS/NA
- o New Authoritative Border Router option
- o New Address Registration Option
- o Separated Prefix Information and Content Information
- o Optional DAD to the edge

Changes from -07 to -08:

- o Removed Extended LoWPAN and Whiteboard related sections.
- o Included reference to the autoconf addressing model.
- o Added Optimistic Flag to 6AO.

- o Added guidelines on routers performing DAD.
- o Removed the NR/NC Advertising Interval.
- o Added assumption of uniform IID formation and DAD throughout a LoWPAN.

Changes from -06 to -07:

- o Updated addressing and address resolution (#60).
- o Changed the Address Option to 6LoWPAN Address Option, fixed S values (#61).
- o Added support for classic RFC4861 RA Prefix Information messages to be processed (#62).
- o Added a section on using 6LoWPAN-ND under a hard-wired RFC4861 stack (#63).
- o Updated the NR/NC message with a new Router flag, combined the Code and Status fields into one byte, and added the capability to carry 6IOs (#64).
- o Made co-existence with other ND mechanisms clear (#59).
- o Added a new Protocol Specification section with all mechanisms specified there (#59).
- o Removed dependencies and conflicts with RFC4861 wherever possible (#59).
- o Some editorial cleanup.

Changes from -05 to -06:

- o Fixed the Prf codes (#52).
- o Corrected the OIIIO TID field to 8-bits. Changed the Nonce/OII order in both the OIIIO and the NR/NC. (#53)
- o Corrected an error in Table 1 (#54).
- o Fixed asymmetric and a misplaced transient in the 6LoWPAN terminology section.
- o Added Updates RFC4861 to header

Changes from -04 to -05:

- o Meaning of the RA's M-bit changed to original [RFC4861] meaning (#46).
- o Terms "on-link" and "off-link" used in place of "on-link" and "off-link".
- o Next-hop determination text simplified (#49).
- o Neighbor cache and destination cache removed.
- o IID to link-layer address requirement relaxed.
- o NR/NC changes to enable on-link refresh with routers (#48).
- o Modified 6LoWPAN Information Option (#47).
- o Added a Protocol Constants section (#24)
- o Added the NR processing table (#51)
- o Considered the use of SeND on backbone NS/NA messages (#50)

Changes from -03 to -04:

- o Moved Ad-hoc LoWPAN operation to Section 7 and made ULA prefix generation a features useful also in Simple and Extended LoWPANs. (#41)
- o Added a 32-bit Owner Nonce to the NR/NC messages and the Whiteboard, removed the TID history. (#39)
- o Improved the duplicate OII detection algorithm using the Owner Nonce. (#39)
- o Clarified the use of Source and Target link-layer options in NR/NC. (#43)
- o Included text on the use of alternative methods to acquire addresses. (#38)
- o Removed S=2 from Address Option (not needed). (#36)
- o Added a section on router dissemination consistency. (#44)
- o Small improvements and extensive editing. (#42, #37, #35)

Changes from -02 to -03:

- o Updated terminology, with RFC4861 non-transitive link model.
- o 6LoWPAN and ND terminology separated.
- o Protocol overview explains RFC4861 diff in detail.
- o RR/RC is now Node Registration/Confirmation (NR/NC).
- o Added NR failure codes.
- o ER Metric now included in 6LoWPAN Summary Option for use in default router determination by hosts.
- o Examples of host data structures, and the Whiteboard given.
- o Whiteboard is supported by all Edge Routers for option simplicity.
- o Edge Router Specification chapter re-structured, clarifying optional Extended LoWPAN operation.
- o NS/NA now completely optional for nodes. No address resolution or NS/NA NUD required.
- o link-local operation now compatible with oDAD (was broken).
- o Exception to hop limit = 255 for NR/NC messages.
- o Security considerations improved.
- o ICMPv6 destination unreachable supported.

Changes from -01 to -02:

- o Fixed 16 != 0xff bug (ticket closed).
- o Specified use of ULAs in ad-hoc LoWPAN section 9 (ticket closed).
- o Terminology cleanup based on Alex's comments.
- o General editing improvements.

Changes from -00 to -01:

- o Specified the duplicate owner interface identifier procedures. A TID lollipop algorithm was sufficient (nonce unnecessary).
- o Defined fault tolerance using secondary bindings.
- o Defined ad-hoc network operation.
- o Removed the E flag from RA and the X flag from RR/RC.
- o Completed message examples.
- o Lots of improvements in text quality and consistency were made.

16. References

16.1. Normative References

- [EUI64] "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", <<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>>.
- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.
- [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.
- [RFC2491] Armitage, G., Schuster, P., Jork, M., and G. Harter, "IPv6 over Non-Broadcast Multiple Access (NBMA) networks", RFC 2491, January 1999.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC5889] Baccelli, E. and M. Townsley, "IP Addressing Model in Ad Hoc Networks", RFC 5889, September 2010.

16.2. Informative References

- [I-D.ietf-6lowpan-hc] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams in 6LoWPAN Networks", draft-ietf-6lowpan-hc-13 (work in progress), September 2010.
- [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

Authors' Addresses

Zach Shelby (editor)
Sensinode
Hallituskatu 13-17D
Oulu 90100
FINLAND

Phone: +358407796297
Email: zach@sensinode.com

Samita Chakrabarti
IP Infusion
1188 Arquest Street
Sunnyvale, CA
USA

Email: samitac@ipinfusion.com

Erik Nordmark
Oracle, Inc.
17 Network Circle
Menlo Park, CA 94025
USA

Email: Erik.Nordmark@Oracle.COM

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 27, 2011

R. Moskowitz
ICSA labs
July 26, 2010

HIP Diet EXchange (DEX)
draft-moskowitz-hip-rg-dex-02

Abstract

This document specifies the details of the Host Identity Protocol Diet EXchange (HIP DEX). HIP DEX is a variant of the HIP Base EXchange (HIP BEX) [RFC5201-bis] specifically designed to use as few crypto primitives as possible yet still deliver the same class of security features as HIP BEX.

The design goal of HIP DEX is to be usable by sensor devices that are code and processor constrained. Like HIP BEX it is expected to be used together with another suitable security protocol, such as the Encapsulated Security Payload (ESP). HIP DEX can also be used directly as a keying mechanism for a MAC layer security protocol as is supported by IEEE 802.15.4 [IEEE.802-15-4.2006].

Status of This Memo

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

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

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

This Internet-Draft will expire on January 27, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
1.1. The HIP Diet EXchange (DEX)	4
1.2. Memo Structure	5
2. Terms and Definitions	6
2.1. Requirements Terminology	6
2.2. Notation	6
3. The DEX Host Identifier Tag (HIT) and Its Representations . .	6
3.1. Host Identity Tag (HIT)	6
3.2. Generating a HIT from an HI	7
4. Protocol Overview	7
4.1. Creating a HIP Association	7
4.1.1. HIP Puzzle Mechanism	8
4.1.2. Puzzle Exchange	9
4.1.3. HIP State Machine	10
4.1.4. HIP DEX Security Associations	14
4.1.5. User Data Considerations	14
5. Packet Formats	15
5.1. HIP Parameters	15
5.1.1. HIT_SUITE_LIST	15
5.1.2. ENCRYPTED_KEY	16
5.1.3. HIP_MAC_3	17
5.2. HIP Packets	17
5.2.1. I1 - the HIP Initiator Packet	18
5.2.2. R1 - the HIP Responder Packet	19
5.2.3. I2 - the Second HIP Initiator Packet	20
5.2.4. R2 - the Second HIP Responder Packet	21

5.3. ICMP Messages	22
6. Packet Processing	22
6.1. Solving the Puzzle	23
6.2. HIP_MAC Calculation and Verification	24
6.2.1. CMAC Calculation	24
6.3. HIP DEX KEYMAT Generation	25
6.4. Processing Incoming I1 Packets	28
6.4.1. R1 Management	28
6.5. Processing Incoming R1 Packets	28
6.6. Processing Incoming I2 Packets	29
6.7. Processing Incoming R2 Packets	30
6.8. Sending UPDATE Packets	30
6.9. Handling State Loss	30
7. HIP Policies	31
8. Security Considerations	31
9. IANA Considerations	32
10. Acknowledgments	32
11. References	32
11.1. Normative References	32
11.2. Informative References	33
Appendix A. Using Responder Puzzles	34
Appendix B. Generating a Public Key Encoding from an HI	35

1. Introduction

NOTE: This version of the draft was prepared for the HIP RG meeting with MOST of the additions worked out since the last draft. However, it was NOT carefully edited and some parts are still mis-matched between the old and new discriptive text.

This memo specifies the details of the Host Identity Protocol Diet EXchange (HIP DEX). HIP DEX uses the smallest possible set of established cryptographic primitives, in such a manner that does not change our understanding of their behaviour, yet in a different formulation to achieve assertions normally met with different primitives.

HIP DEX builds on HIP BEX [RFC5201-bis], and only the differences between BEX and DEX are documented here.

There are a few key differences between BEX and DEX.

- Minimum collection of cryptographic primitives.

 - AES-CBC for symmetric encryption and to provide CMAC for MACing functions.

 - Static/Static Elliptic Curve Diffie-Hellman keys used to encrypt the session key.

 - A simple truncation function for HIT generation.

- Forfeit of Perfect Forward Secrecy with the dropping of ephemeral Diffie-Hellman.

- Forfeit of digital signatures with the removal of a hash function. Reliance of DH derived key used in HIP_MAC to prove ownership of the private key.

- Provide a Password Authentication within the exchange. This may be supported by BEX as well, but not defined there.

- Operate in an aggressive retransmission manner to deal with the high packet loss nature of sensor networks. This retransmission also provides an indirect acknowledgement of exchange completion.

1.1. The HIP Diet EXchange (DEX)

The HIP diet exchange is a two-party cryptographic protocol used to establish communications context between hosts. The first party is called the Initiator and the second party the Responder. The four-

packet design helps to make HIP DoS resilient. The protocol transmits an EC Diffie-Hellman encrypted key in the 3rd and 4th packets, and authenticates the parties also in the 3rd and 4th packets. Additionally, the Responder starts a puzzle exchange in the 2nd packet, with the Initiator completing it in the 3rd packet before the Responder stores any state from the exchange.

Thus DEX is operationally similar to BEX, just keyed more along the lines of TLS. The model is also fairly equivalent to 802.11-2007 [IEEE.802-11.2007] Master Key and Pair-wise Transient Key, but handled in a single exchange.

HIP DEX does not have the option of encrypting the Host Identity of the Initiator in the 3rd packet. The Responder's Host Identity is also not protected. Thus there is no attempt at anonymity as in BEX.

Data packets start to flow after the 4th packet. HIP DEX does not have an explicit transition for the Responder to connected state. This is learned when the Responder starts receiving protected datagrams, indicating that the Initiator received the R2 packet. As such the Initiator should take care to NOT send the first data packet until some delta time after it received the R2 packet. This is to provide time for the Responder to process any aggressively retransmitted I2 packets.

An existing HIP association can be updated using the update mechanism defined in this document, and when the association is no longer needed, it can be closed using the defined closing mechanism.

Finally, HIP is designed as an end-to-end authentication and key establishment protocol, to be used with Encapsulated Security Payload (ESP) [RFC5202] and other end-to-end security protocols. The base protocol does not cover all the fine-grained policy control found in Internet Key Exchange (IKE) [RFC4306] that allows IKE to support complex gateway policies. Thus, HIP is not a replacement for IKE.

1.2. Memo Structure

The rest of this memo is structured as follows. Section 2 defines the central keywords, notation, and terms used throughout the rest of the document. Section 4 gives an overview of the HIP base exchange protocol. Section 6 defines the rules for packet processing. Finally, Sections 7, 8, and 9 discuss policy, security, and IANA considerations, respectively.

2. Terms and Definitions

2.1. Requirements Terminology

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

2.2. Notation

[x] indicates that x is optional.

{x} indicates that x is encrypted.

X(y) indicates that y is a parameter of X.

<x>i indicates that x exists i times.

--> signifies "Initiator to Responder" communication (requests).

<-- signifies "Responder to Initiator" communication (replies).

| signifies concatenation of information-- e.g., X | Y is the concatenation of X with Y.

Ltrunc (M(x), K) denotes the lowest order K bits of the result of the mac function M on the input x.

3. The DEX Host Identifier Tag (HIT) and Its Representations

The DEX Host Identity Tag (HIT) is distinguished in two ways from the BEX HIT:

The HIT SUITE ID Section 5.1.1 is ONLY a DEX ID.

The HIT DEX hit is not generated via a cryptographic hash. Rather it is a truncation of the Elliptic Curve Host Identity.

3.1. Host Identity Tag (HIT)

The DEX Host Identity Tag is a 128-bit value -- a truncation of the Host Identifier. There are two advantages of using a Host Identity Tag over the actual Host Identity public key in protocols. Firstly, its fixed length makes for easier protocol coding and also better manages the packet size cost of this technology. Secondly, it presents a consistent format to the protocol whatever underlying identity technology is used.

BEX uses RFC 4843-bis [RFC4843-bis] specified 128-bit hash-based identifiers, called Overlay Routable Cryptographic Hash Identifiers (ORCHIDs). Their prefix, allocated from the IPv6 address block, is defined in [RFC4843-bis].

In DEX, a cryptographic hash is NOT used to form the HIT. Rather the HI is truncated to 96 bits.

3.2. Generating a HIT from an HI

The DEX HIT is not an ORCHID, as there is no hash function in DEX. Since a HI that is an ECDH key is directly computed from a random number it is already collision resistant. The DEX HIT is the left-truncated 96 bits of the HI. This 96 bit value is used in place of the hash in the ORCHID. The HIT suite (see Section 9) is used for the four bits of the Orchid Generation Algorithm (OGA) field in the ORCHID. The same IPv6 prefix used in BEX is used for DEX.

4. Protocol Overview

The following material is an overview of the differences between the BEX and DEX implementations of the HIP protocol. It is expected that [RFC5201-bis] is well understood first.

4.1. Creating a HIP Association

By definition, the system initiating a HIP exchange is the Initiator, and the peer is the Responder. This distinction is forgotten once the base exchange completes, and either party can become the Initiator in future communications.

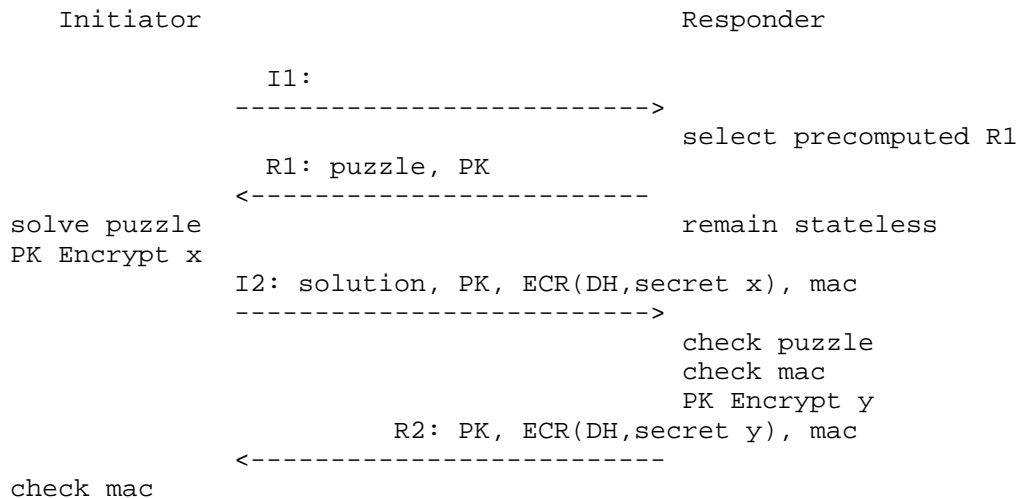
The HIP Diet EXchange serves to manage the establishment of state between an Initiator and a Responder. The first packet, I1, initiates the exchange, and the last three packets, R1, I2, and R2, constitute an authenticated secret key wrapped by a Diffie-Hellman derived key for session key generation. The HIP association keys are drawn from this keying material. If other cryptographic keys are needed, e.g., to be used with ESP, they are expected to be drawn from the same keying material.

The second packet, R1, starts the actual exchange. It contains a puzzle -- a cryptographic challenge that the Initiator must solve before continuing the exchange. The level of difficulty of the puzzle can be adjusted based on level of trust with the Initiator, current load, or other factors. The R1 also contains lists of cryptographic algorithms supported by the Responder. Based on these lists, the Initiator can continue, abort, or restart the base exchange with a different selection of cryptographic algorithms.

In the I2 packet, the Initiator must display the solution to the received puzzle. Without a correct solution, the I2 message is discarded. The I2 also contains a key wrap parameter that carries the key for the Responder. This key is only half the final session key. The packet is MACed by the sender (Initiator).

The R2 packet finalizes the base exchange. The R2 contains a key wrap parameter that carries the rest of the key for the Initiator. The packet is MACed by the sender (Initiator).

The base exchange is illustrated below. The term "key" refers to the Host Identity public key, "secret" refers to a random value encrypted by a public key, and "sig" represents a signature using such a key. The packets contain other parameters not shown in this figure.



4.1.1. HIP Puzzle Mechanism

The purpose of the HIP puzzle mechanism is to protect the Responder from a number of denial-of-service threats. It allows the Responder to delay state creation until receiving I2. Furthermore, the puzzle allows the Responder to use a fairly cheap calculation to check that the Initiator is "sincere" in the sense that it has churned CPU cycles in solving the puzzle.

DEX uses the CMAC function instead of a hash function as in BEX.

The puzzle mechanism has been explicitly designed to give space for various implementation options. It allows a Responder implementation to completely delay session-specific state creation until a valid I2

is received. In such a case, a correctly formatted I2 can be rejected only once the Responder has checked its validity by computing one CMAC function. On the other hand, the design also allows a Responder implementation to keep state about received I1s, and match the received I2s against the state, thereby allowing the implementation to avoid the computational cost of the CMAC function. The drawback of this latter approach is the requirement of creating state. Finally, it also allows an implementation to use other combinations of the space-saving and computation-saving mechanisms.

Generally speaking, the puzzle mechanism works in DEX the same as in BEX. There are some implementation differences, using CMAC rather than a hash.

See Appendix A for one possible implementation. Implementations SHOULD include sufficient randomness to the algorithm so that algorithmic complexity attacks become impossible [CRO03].

4.1.2. Puzzle Exchange

The Responder starts the puzzle exchange when it receives an I1. The Responder supplies a random number I, and requires the Initiator to find a number J. To select a proper J, the Initiator must create the concatenation of the HITs of the parties and J, and feed this concatenation using I as the key into the CMAC algorithm. The lowest order K bits of the result MUST be zeros. The value K sets the difficulty of the puzzle.

To generate a proper number J, the Initiator will have to generate a number of Js until one produces the CMAC target of zeros. The Initiator SHOULD give up after exceeding the puzzle lifetime in the PUZZLE parameter ([RFC5201-bis]). The Responder needs to re-create the concatenation of the HITs and the provided J, and compute the CMAC using I once to prove that the Initiator did its assigned task.

To prevent precomputation attacks, the Responder MUST select the number I in such a way that the Initiator cannot guess it. Furthermore, the construction MUST allow the Responder to verify that the value was indeed selected by it and not by the Initiator. See Appendix A for an example on how to implement this.

Using the Opaque data field in an ECHO_REQUEST_UNSIGNED parameter ([RFC5201-bis]), the Responder can include some data in R1 that the Initiator must copy unmodified in the corresponding I2 packet. The Responder can generate the Opaque data in various ways; e.g., using some secret, the sent I, and possibly other related data. Using the same secret, the received I (from the I2), and the other related data (if any), the Receiver can verify that it has itself sent the I to

the Initiator. The Responder MUST periodically change such a used secret.

It is RECOMMENDED that the Responder generates a new puzzle and a new R1 once every few minutes. Furthermore, it is RECOMMENDED that the Responder remembers an old puzzle at least 2*Lifetime seconds after the puzzle has been deprecated. These time values allow a slower Initiator to solve the puzzle while limiting the usability that an old, solved puzzle has to an attacker.

4.1.3. HIP State Machine

The HIP protocol itself has little state. In HIP DEX, as in BEX, there is an Initiator and a Responder. Once the security associations (SAs) are established, this distinction is lost. If the HIP state needs to be re-established, the controlling parameters are which peer still has state and which has a datagram to send to its peer.

The HIP DEX state machine has the same states as the BEX state machine. However, there is an optional to implement aggressive transmission feature to provide better performance in sensor networks with high packet loss. the following documents the few differences in the DEX state machine.

4.1.3.1. HIP Aggressive Transmission Mechanism

HIP DEX may be used on networks with high packet loss. DEX deals with this by using an aggressive transmission practice for I1 and I2 packets. The Initiator SHOULD continually send I1 and I2 packets at some short interval t msec, based on local policy. The transmission stops on receipt of the corresponding R1 or R2 packet, which acts as an acknowledgment receipt.

Since the Responder is stateless until it receives an I2, it does not need any special behaviour on sending R1 other than to send one whenever it receives an I1. The Responder sends an R2 after receipt every I2. The Responder does need to know that R2 was received by the Initiator. Like in BEX, the Responder can learn this when it starts receiving datagrams.

4.1.3.2. HIP States

State	Explanation
UNASSOCIATED	State machine start
I1-SENT	Initiating base exchange
I2-SENT	Waiting to complete base exchange
R2-SENT	Waiting to complete base exchange
ESTABLISHED	HIP association established
CLOSING	HIP association closing, no data can be sent
CLOSED	HIP association closed, no data can be sent
E-FAILED	HIP exchange failed

Table 1: HIP States

4.1.3.3. HIP State Processes

System behavior in state I1-SENT, Table 2.

Trigger	Action
t msec	Send I1 and stay at I1-SENT

Table 2: I1-SENT - Initiating HIP

System behavior in state I2-SENT, Table 3.

Trigger	Action
t msec	Send I2 and stay at I2-SENT

Table 3: I2-SENT - Waiting to finish HIP

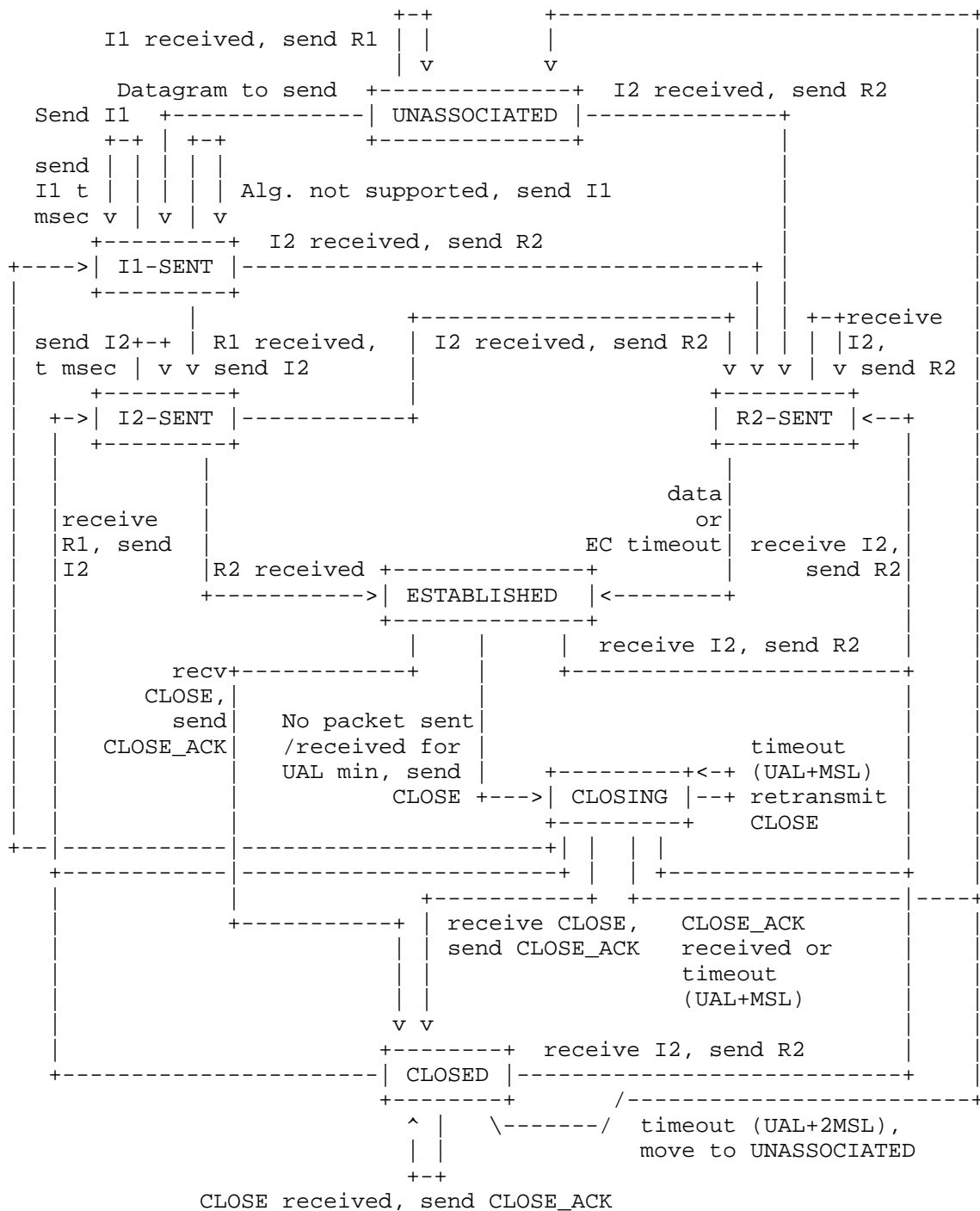
System behavior in state R2-SENT, Table 4.

Trigger	Action
Receive duplicate I2	Send R2 and stay at R2-SENT

Table 4: R2-SENT - Waiting to finish HIP

4.1.3.4. Simplified HIP State Diagram

The following diagram shows the major state transitions. Transitions based on received packets implicitly assume that the packets are successfully authenticated or processed.



4.1.4. HIP DEX Security Associations

HIP DEX establishes two Security Associations (SA), one for the Diffie-Hellman derived key, or Master Key, and one for session or Pair-wise Key.

4.1.4.1. Master Key SA

The Master Key SA is used to secure DEX parameters and authenticate HIP packets. Since so little data will be protected by this SA it can be very long lived.

The Master Key SA contains the following elements.

Source HIT

Destination HIT

HIP_Encrypt Key

HIP_MAC Key

Both keys are extracted from the Diffie-Hellman derived key via Section 6.3. Their length is determined by HIP_CIPHER.

4.1.4.2. Pair-wise Key SA

The Pair-wise Key SA is used to secure and authenticate user data. It is refreshed (or rekeyed) using the UPDATE packet exchange.

The Pair-wise Key SA elements are defined by the data transform (e.g. ESP_TRANSFORM [RFC5202]).

The secrets in ENCRYPTED_KEY from I2 and R2 are concatenated to form the input to a Key Derivation Function (KDF). If the data transform does not have its own KDF, then Section 6.3 is used. Even though this input is randomly distributed, a KDF Extract phase may be needed to get the proper length for input to the KDF Expand phase.

4.1.5. User Data Considerations

There is no difference in User Data Considerations between BEX and DEX with one exception. Loss of state due to system reboot may be a critical performance issue. Thus implementors MAY choose to use non-volatile, secure storage for HIP state so it will survive system reboot. This will limit state loss during reboots to only those situations that there is an SA timeout.

5. Packet Formats

5.1. HIP Parameters

The HIP Parameters are used to carry the public key associated with the sender's HIT, together with related security and other information. They consist of ordered parameters, encoded in TLV format.

The following new parameter types are currently defined for DEX, in addition to those defined for BEX. Also listed are BEX parameters that have additional values for DEX.

For the BEX parameters, `DIFFIE_HELMAN`, `DH_GROUP_LIST`, and `HOST_ID`, only the ECC values are valid in DEX.

TLV	Type	Length	Data
ENCRYPTED_KEY	643	variable	Encrypted container of for key generation exchange
HIP_MAC_3	61507	variable	CMAC-based message authentication code
HIT_SUITE_LIST	715	variable	Ordered list of the HIT suites supported by the Responder

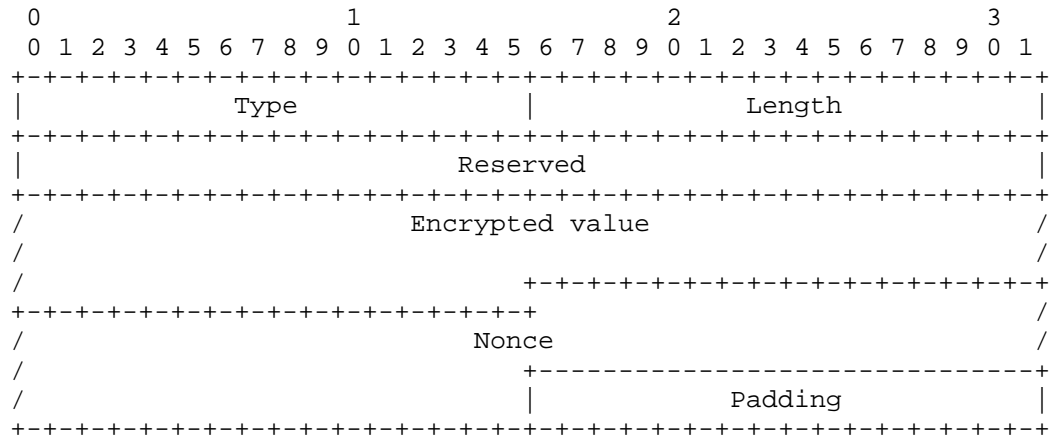
5.1.1. HIT_SUITE_LIST

The HIT suites in DEX are limited to:

HIT suite	ID
ECDH/DEX	8

The `HIT_SUITE_LIST` parameter contains a list of the supported HIT suite IDs of the Responder. Since the HIT of the Initiator is a DEX HIT, the Responder MUST only respond with a DEX HIT suite ID. Currently, only one such suite ID has been defined.

5.1.2. ENCRYPTED_KEY



Type 643

Length length in octets, excluding Type, Length, and
Padding

Encrypted
value The value is encrypted using an encryption algorithm
as defined in the HIP_CIPHER parameter.

Nonce Nonce included in encrypted text.

The ENCRYPTED parameter encapsulates a value and a nonce. The value is typically a random number used in a key creation process and the nonce is known to the receiver to validate successful decryption.

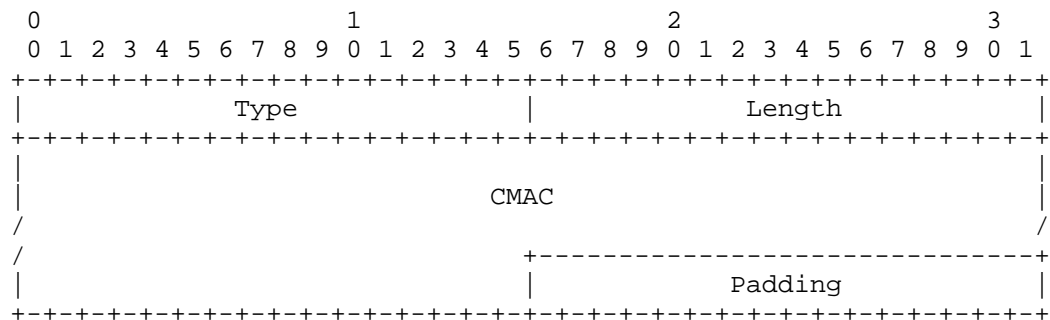
Some encryption algorithms require an IV (initialization vector). The IV MUST be known to the receiver through some source other than within the Encrypted_key block. For example the Puzzle value, I, can be used as an IV.

Some encryption algorithms require that the data to be encrypted must be a multiple of the cipher algorithm block size. In this case, the above block of data MUST include additional padding, as specified by the encryption algorithm. The size of the extra padding is selected so that the length of the unencrypted data block is a multiple of the cipher block size. The encryption algorithm may specify padding bytes other than zero; for example, AES [FIPS.197.2001] uses the PKCS5 padding scheme (see section 6.1.1 of [RFC2898]) where the remaining n bytes to fill the block each have the value n. This yields an "unencrypted data" block that is transformed to an "encrypted data" block by the cipher suite. This extra padding added to the set of parameters to satisfy the cipher block alignment rules is not counted in HIP TLV length fields, and this extra padding should be removed by the cipher suite upon decryption.

Note that the length of the cipher suite output may be smaller or larger than the length of the value and nonce to be encrypted, since the encryption process may compress the data or add additional padding to the data.

Once this encryption process is completed, the Encrypted_key data field is ready for inclusion in the Parameter. If necessary, additional Padding for 8-byte alignment is then added according to the rules of TLV Format in [RFC5201-bis].

5.1.3. HIP_MAC_3



Type	61507
Length	length in octets, excluding Type, Length, and Padding
CMAC	CMAC computed over the HIP packet, excluding the HIP_MAC parameter itself. The checksum field MUST be set to zero and the HIP header length in the HIP common header MUST be calculated not to cover any excluded parameters when the CMAC is calculated. The size of the CMAC is the natural size of the AES block depending on the AES key size.

The CMAC calculation and verification process is presented in Section 6.2.1.

5.2. HIP Packets

DEX uses the same eight basic HIP packets (see [RFC5201-bis]) as BEX. Four are for the HIP exchange, one is for updating, one is for sending notifications, and two are for closing a HIP association. There are some differences in the HIP parameters in the exchange packets between BEX and DEX. This section will cover the DEX packets.

An important difference between BEX and DEX HIP packets is that there

is NO HIP_SIGNATURE available in DEX. Thus R1 is completely unprotected and can be spoof. The I2, R2, UPDATE, NOTIFY, CLOSE, and CLOSE_ACK only have HIP_MAC_3 for packet authentication. The processing of these packets are changed accordingly.

In the future, an OPTIONAL upper-layer payload MAY follow the HIP header. The Next Header field in the header indicates if there is additional data following the HIP header. The HIP packet, however, MUST NOT be fragmented. This limits the size of the possible additional data in the packet.

5.2.1. I1 - the HIP Initiator Packet

The HIP header values for the I1 packet:

Header:

Packet Type = 1

SRC HIT = Initiator's HIT

DST HIT = Responder's HIT, or NULL

IP (HIP (DH_GROUP_LIST))

Minimum size = 40 bytes

The I1 packet contains the fixed HIP header and the Initiator's DH_GROUP_LIST.

Valid control bits: none

The Initiator HIT MUST be a DEX HIT. That is the HIT Suite ID MUST be of a DEX type. Currently only ECDH/DEX is defined.

The Initiator gets the Responder's HIT either from a DNS lookup of the Responder's FQDN, from some other repository, or from a local table. The Responder's HIT MUST be a DEX HIT. If the Initiator does not know the Responder's HIT, it may attempt to use opportunistic mode by using NULL (all zeros) as the Responder's HIT. See also "HIP Opportunistic Mode" [RFC5201-bis].

Since this packet is so easy to spoof even if it were signed, no attempt is made to add to its generation or processing cost.

The Initiator includes a DH_GROUP_LIST parameter in the I1 to inform the Responder of its preferred DH Group IDs. Only ECDH Groups may be included in this list. Note that the DH_GROUP_LIST in the I1 packet is not protected by a MAC.

Implementations MUST be able to handle a storm of received I1

packets, discarding those with common content that arrive within a small time delta, but distinguishing this from arriving at a set time delta. This behaviour is the expected behaviour for an Initiator on a network with high packet loss. The HIP state machine calls out this behaviour in this case and the Initiator will stop sending I1 packets after it receives an R1 packet.

5.2.2. R1 - the HIP Responder Packet

The HIP header values for the R1 packet:

Header:

Packet Type = 2
SRC HIT = Responder's HIT
DST HIT = Initiator's HIT

```
IP ( HIP ( [ R1_COUNTER, ]
           PUZZLE,
           HIP_CIPHER,
           HOST_ID,
           HIT_SUITE_LIST,
           DH_GROUP_LIST,
           [ <, ECHO_REQUEST_UNSIGNED >i ] )
```

Minimum size = 120 bytes

Valid control bits: A

If the Responder's HI is an anonymous one, the A control MUST be set.

The Initiator's HIT MUST match the one received in I1. If the Responder has multiple HIs, the Responder's HIT used MUST match Initiator's request. If the Initiator used opportunistic mode, the Responder may select freely among its HIs. See also "HIP Opportunistic Mode" [RFC5201-bis].

The R1 generation counter is used to determine the currently valid generation of puzzles. The value is increased periodically, and it is RECOMMENDED that it is increased at least as often as solutions to old puzzles are no longer accepted.

The Puzzle contains a Random #I and the difficulty K. The difficulty K indicates the number of lower-order bits, in the puzzle CMAC result, that must be zeros; see Section 4.1.2.

The Initiator HIT does not provide the HOST_ID key size. The Responder selects its HOST_ID based on the Initiator's preference expressed in the DH_GROUP_LIST parameter in the I1. The Responder

sends back its own preference based on which it chose the HOST_ID as DH_GROUP_LIST. This allows the Initiator to determine whether its own DH_GROUP_LIST in the I1 was manipulated by an attacker. There is a further risk that the Responder's DH_GROUP_LIST was manipulated by an attacker, as R1 cannot be authenticated in DEX as it can in BEX. Thus it is repeated in R2 allowing for a final check at that point.

In DEX, the Diffie-Hellman HOST_ID values are static. They are NOT discarded.

The HIP_CIPHER contains the encryption algorithms supported by the Responder to protect the key exchange, in the order of preference. All implementations MUST support the AES-CBC [RFC3602].

The ECHO_REQUEST_UNSIGNED contains data that the sender wants to receive unmodified in the corresponding response packet in the ECHO_RESPONSE_UNSIGNED parameter.

5.2.3. I2 - the Second HIP Initiator Packet

The HIP header values for the I2 packet:

Header:

Type = 3

SRC HIT = Initiator's HIT

DST HIT = Responder's HIT

```
IP ( HIP ( [R1_COUNTER,]
           SOLUTION,
           HIP_CIPHER,
           HOST_ID,
           ENCRYPTED_KEY {DH, secret-x|I},
           [ ENCRYPTED {DH, ENCRYPTED_KEY {passwd, challenge } },]
           HIP_MAC_3,
           [<, ECHO_RESPONSE_UNSIGNED>i ] ) )
```

Minimum size = 180 bytes

Valid control bits: A

The HITs used MUST match the ones used previously.

If the Initiator's HI is an anonymous one, the A control MUST be set.

The Initiator MAY include an unmodified copy of the R1_COUNTER parameter received in the corresponding R1 packet into the I2 packet.

The Solution contains the Random #I from R1 and the computed #J. The

low-order K bits of the CMAC(S, | ... | J) MUST be zero.

In DEX, the Diffie-Hellman HOST_ID values are static. They are NOT discarded.

The HIP_CIPHER contains the single encryption transform selected by the Initiator, that will be used to protect the HI exchange. The chosen transform MUST correspond to one offered by the Responder in the R1. All implementations MUST support the AES-CBC transform [RFC3602].

The ECHO_RESPONSE_UNSIGNED contain the unmodified Opaque data copied from the corresponding echo request parameter.

The ENCRYPTED_KEY contains an Initiator generated random secret x that MUST be uniformly distributed that is concatenated with I from the puzzle. The secret x's length matches the keysize of the selected encryption transform. I from the puzzle is used as the IV in the encryption transform. This acts as a nonce from the Responder to prove freshness of the secret wrapping from the Initiator. I in the ENCRYPTED block enables the Responder to validate a proper decryption of the block. The key for the encryption is the HIP_Encrypt key.

If the Initiator has prior knowledge that the Responder is expecting a password authentication, the Initiator encrypts the ECHO_REQUEST_UNSIGNED with the password, then wraps the ENCRYPTED parameter in the secret x. I from the puzzle is used as the nonce here as well. There is no signal within R1 for this behaviour. Knowledge of password authentication must be externally configured.

The MAC is calculated over the whole HIP envelope, excluding any parameters after the HIP_MAC_3, as described in Section 6.2.1. The Responder MUST validate the HIP_MAC_3.

5.2.4. R2 - the Second HIP Responder Packet

The HIP header values for the R2 packet:

Header:

Packet Type = 4
SRC HIT = Responder's HIT
DST HIT = Initiator's HIT

IP (HIP (DH_GROUP_LIST,
 ENCRYPTED_KEY {DH, secret-y|I},
 HIP_MAC_3)

Minimum size = 108 bytes

Valid control bits: none

The Responder repeats the DH_GROUP_LIST parameter in R2. This MUST be the same list as included in R1. The DH_GROUP_LIST parameter is repeated here because R2 is MACed and thus cannot be altered by an attacker. This allows the Initiator to determine whether its own DH_GROUP_LIST in the I1 was manipulated by an attacker.

The ENCRYPTED contains an Responder generated random secret y that MUST be uniformly distributed that is concatenated with I from the puzzle. The secret y 's length matches the keysize of the selected encryption transform. I from the puzzle is used as the IV in the encryption transform. This acts as a nonce from the Initiator to prove freshness of the secret wrapping from the Responder. I in the ENCRYPTED block enables the Responder to validate a proper decryption of the block. The key for the encryption is the HIP_Encrypt key.

The HIP_MAC_3 is calculated over the whole HIP envelope, with Responder's HOST_ID parameter concatenated with the HIP envelope. The HOST_ID parameter is removed after the CMAC calculation. The procedure is described in Section 6.2.1.

The Initiator MUST validate the HIP_MAC_3.

5.3. ICMP Messages

When a HIP implementation detects a problem with an incoming packet, and it either cannot determine the identity of the sender of the packet or does not have any existing HIP association with the sender of the packet, it MAY respond with an ICMP packet. Any such replies MUST be rate-limited as described in [RFC2463]. In most cases, the ICMP packet will have the Parameter Problem type (12 for ICMPv4, 4 for ICMPv6), with the Pointer field pointing to the field that caused the ICMP message to be generated.

6. Packet Processing

Each host is assumed to have a single HIP protocol implementation that manages the host's HIP associations and handles requests for new ones. Each HIP association is governed by a conceptual state machine, with states defined above in Section 4.1.3. The HIP implementation can simultaneously maintain HIP associations with more than one host. Furthermore, the HIP implementation may have more than one active HIP association with another host; in this case, HIP associations are distinguished by their respective HITs. It is not possible to have more than one HIP association between any given pair

of HITs. Consequently, the only way for two hosts to have more than one parallel association is to use different HITs, at least at one end.

6.1. Solving the Puzzle

This subsection describes the puzzle-solving details.

In R1, the values I and K are sent in network byte order. Similarly, in I2, the values I and J are sent in network byte order. The mac is created by concatenating, in network byte order, the following data, in the following order and using the CMAC algorithm with I as the key:

128-bit Initiator's HIT, in network byte order, as appearing in the HIP Payload in R1 and I2.

128-bit Responder's HIT, in network byte order, as appearing in the HIP Payload in R1 and I2.

n-bit random value J (where n is CMAC-len), in network byte order, as appearing in I2.

In order to be a valid response puzzle, the K low-order bits of the resulting CMAC must be zero.

Notes:

- i) All the data in the CMAC input MUST be in network byte order.
- ii) The order of the Initiator's and Responder's HITs are different in the R1 and I2 packets; see [RFC5201-bis]. Care must be taken to copy the values in the right order to the CMAC input.

The following procedure describes the processing steps involved, assuming that the Responder chooses to precompute the R1 packets:

Precomputation by the Responder:

Sets up the puzzle difficulty K.
Creates a R1 and caches it.

Responder:

Selects a suitable cached R1.
Generates a random number I.
Sends I and K in an R1.
Saves I and K for a Delta time.

Initiator:

Generates repeated attempts to solve the puzzle until a matching J is found:

Ltrunc(CMAC(I, HIT-I | HIT-R | J), K) == 0
Sends I and J in an I2.

Responder:

Verifies that the received I is a saved one.

Finds the right K based on I.

Computes $V := \text{Ltrunc}(\text{CMAC}(I, \text{HIT-I} \mid \text{HIT-R} \mid J), K)$

Rejects if $V \neq 0$

Accept if $V == 0$

6.2. HIP_MAC Calculation and Verification

The following subsections define the actions for processing the HIP_MAC_3 parameter.

6.2.1. CMAC Calculation

Both the Initiator and the Responder should take some care when verifying or calculating the HIP_MAC_3. Specifically, the Responder should preserve other parameters than the HOST_ID when sending the R2. Also, the Initiator has to preserve the HOST_ID exactly as it was received in the R1 packet.

The scope of the calculation for HIP_MAC_3 is:

CMAC: { HIP header | [Parameters] }

where Parameters include all HIP parameters of the packet that is being calculated with Type values from 1 to (HIP_MAC's Type value - 1) and exclude parameters with Type values greater or equal to HIP_MAC's Type value.

During HIP_MAC calculation, the following applies:

- o In the HIP header, the Checksum field is set to zero.
- o In the HIP header, the Header Length field value is calculated to the beginning of the HIP_MAC parameter.

Parameter order is described in [RFC5201-bis].

The HIP_MAC parameter is defined in Section 5.1.3. The CMAC calculation and verification process is as follows:

Packet sender:

1. Create the HIP packet, without the HIP_MAC or any other parameter with greater Type value than the HIP_MAC parameter has.
2. Calculate the Header Length field in the HIP header.
3. Compute the CMAC using either HIP-gl or HIP-lg integrity key retrieved from KEYMAT as defined in Section 6.3.
4. Add the HIP_MAC_3 parameter to the packet and any parameter with greater Type value than the HIP_MAC's (HIP_MAC_3's) that may follow.
5. Recalculate the Length field in the HIP header.

Packet receiver:

1. Verify the HIP header Length field.
2. Remove the HIP_MAC_3 parameter, as well as all other parameters that follow it with greater Type value, saving the contents if they will be needed later.
3. Recalculate the HIP packet length in the HIP header and clear the Checksum field (set it to all zeros).
4. Compute the CMAC using either HIP-gl or HIP-lg integrity key as defined in Section 6.3 and verify it against the received CMAC.
5. Set Checksum and Header Length field in the HIP header to original values.

6.3. HIP DEX KEYMAT Generation

The HIP DEX KEYMAT process is used for both the Diffie-Hellman Derived Master key and the Encrypted secrets Pair-wise key. The former uses both the Extract and Expand phases, while the later MAY need the Extract and Expand phases if the key is longer than 128 bits. Otherwise it only needs the Expand phase.

The Diffie-Hellman Derived Master key is exchanged in R1 and I2 and used in I2, R2. UPDATE, NOTIFY, and ACK packets. The Encrypted secrets Pair-wise key is not used in HIP, but is available as the datagram protection key. Some datagram protection mechanisms have their own Key Derivation Function, and if so that SHOULD be used rather than the HIP DEX KEYMAT.

The KEYMAT has two components, CKDF-Extract and CKDF-Expand. The Extract function COMPRESSES a non-uniformly distributed key, as is the output of a Diffie-Hellman key derivation, to EXTRACT all the key entropy into a fixed length output. The Expand function takes either the output of the Extract function or directly uses a uniformly distributed key and EXPANDS the length of the key, repeatedly distributing the key entropy, to produce the keys needed.

The CKDF-Extract function is following operation; the | operation denotes concatenation.

$$\text{CKDF-Extract}(\text{DHK}, \text{info}, L) \rightarrow \text{CK}$$

where

$$\begin{aligned} \text{info} &= \text{sort}(\text{HIT-I} \mid \text{HIT-R}) \mid \text{"CKDF-Extract"} \\ \text{BigK} &= \text{Diffie-Hellman Derived or Session (x} \mid \text{y) Key} \\ \text{I} &= \text{I from PUZZLE Parameter} \end{aligned}$$

The output CK is calculated as follows:

$$\text{CK} = \text{CMAC}(\text{I}, \text{BigK} \mid \text{info})$$

The CKDF-Expand function is following operation; the | operation denotes concatenation.

CKDF-Expand(CK, info, L) -> OKM

where

```
info    = sort(HIT-I | HIT-R) | "CKDF-Expand"
CK      = CK from CKDF-Extract or (x | y)
PRKlen  = Length of PRK in octets
macLen  = Length of CMAC in octets = 128/8 = 16
L       = length of output keying material in octets
          (<= 255*macLen)
```

If PRKlen != macLen then PRK = CMAC(0¹²⁸, PRK)

The output OKM is calculated as follows:

```
N = ceil(L/macLen)
T = T(1) | T(2) | T(3) | ... | T(N)
OKM = first L octets of T
```

where:

```
T(0) = empty string (zero length)
T(1) = CMAC(CK, T(0) | info | 0x01)
T(2) = CMAC(CK, T(1) | info | 0x02)
T(3) = CMAC(CK, T(2) | info | 0x03)
...
```

(where the constant concatenated to the end of each T(n) is a single octet.)

Sort(HIT-I | HIT-R) is defined as the network byte order concatenation of the two HITs, with the smaller HIT preceding the larger HIT, resulting from the numeric comparison of the two HITs interpreted as positive (unsigned) 128-bit integers in network byte order.

x and y values are from the ENCRYPTED parameters from I2 and R2 respectively.

The initial keys are drawn sequentially in the order that is determined by the numeric comparison of the two HITs, with comparison method described in the previous paragraph. HOST_g denotes the host with the greater HIT value, and HOST_l the host with the lower HIT value.

The drawing order for initial keys:

HIP-gl encryption key for HOST_g's outgoing HIP packets

HIP-gl integrity (CMAC) key for HOST_g's outgoing HIP packets

HIP-lg encryption key (currently unused) for HOST_l's outgoing HIP packets

HIP-lg integrity (CMAC) key for HOST_l's outgoing HIP packets

The number of bits drawn for a given algorithm is the "natural" size of the keys. For the mandatory algorithms, the following sizes apply:

AES 128 or 256 bits

If other key sizes are used, they must be treated as different encryption algorithms and defined separately.

6.4. Processing Incoming I1 Packets

An implementation SHOULD reply to an I1 with an R1 packet, unless the implementation is unable or unwilling to set up a HIP association. An I1 in DEX is handled identically to BEX with the exception that in constructing the R1, the Responder SHOULD select a HIT that is constructed with the MUST algorithm, which is currently ECDH.

6.4.1. R1 Management

All compliant implementations MUST produce R1 packets. An R1 in DEX is handled identically to BEX.

6.5. Processing Incoming R1 Packets

A system receiving an R1 MUST first check to see if it has sent an I1 to the originator of the R1 (i.e., it is in state I1-SENT). An R1 in DEX is handled identically to BEX with the following differences.

If the system has been sending out a stream of I1 packets to work around high packet loss on a network, it stops sending the I1 packets AFTER successfully processing the R1 packet.

There is no HIP_SIGNATURE on this packet. This is an unauthentication packet.

The following steps define the conceptual processing rules for responding to an R1 packet that are different than in BEX:

1. If the system is configured with a authentication password for the responder, it constructs the authentication response to include in the I2.
2. The system prepares and sends an I2, as described in Section 5.2.3. The system MAY be configured to continually send this I2 until it receives and validates an R2.

6.6. Processing Incoming I2 Packets

Upon receipt of an I2, the system MAY perform initial checks to determine whether the I2 corresponds to a recent R1 that has been sent out, if the Responder keeps such state. An I2 in DEX is handled identically to BEX with the following differences.

The HIP implementation SHOULD process the I2. This includes validation of the puzzle solution, extracting the ENCRYPTED key for processing I2, decrypting the Initiator's Host Identity, verifying the mac, creating state, and finally sending an R2.

There is no HIP_SIGNATURE on this packet. Authentication is completely based on the HIP_MAC_3 parameter.

The following steps define the conceptual processing rules for responding to an I2 packet:

1. If the system's state machine is in the I2-SENT state, the system makes a comparison between its local and sender's HITs (similarly as in Section 6.3). If the local HIT is smaller than the sender's HIT, it should drop the I2 packet, and continue using the R1 received and I2 sent to the peer earlier. Otherwise, the system should process the received I2 packet and drop any previously derived Diffie-Hellman keying material K_{ij} and ENCRYPTED keying material it might have formed upon sending the I2 previously. The peer Diffie-Hellman key, ENCRYPTED keying material and the nonce J are taken from the just arrived I2 packet. The local Diffie-Hellman key and the nonce I are the ones that were earlier sent in the R1 packet.
2. The system MUST validate the solution to the puzzle by computing the mac described in Section 5.2.3 using the CMAC algorithm.
3. The system must extract the keying material from the ENCRYPTED parameter. This key is used to derive the HIP data keys.
4. If the checks above are valid, then the system proceeds with further I2 processing; otherwise, it discards the I2 and its state machine remains in the same state. If the system has been

sending a stream of R1 packets to the HIT in the I2 the system stops sending the R1s.

6.7. Processing Incoming R2 Packets

An R2 received in states UNASSOCIATED, I1-SENT, or ESTABLISHED results in the R2 being dropped and the state machine staying in the same state. If an R2 is received in state I2-SENT, it SHOULD be processed.

There is no HIP_SIGNATURE on this packet. Authentication is completely based on the HIP_MAC_3 parameter.

The conceptual processing rules for an incoming R2 packet in DEX are identical to BEX with the following differences.

1. The system checks the DH_GROUP_LIST as in R1 packet processing. If the list is different from R1's there may have been a DH downgrade attack against the unprotected R1 packet. If the DH_GROUP_LIST presents a better list than recieved in the R1 packet, the system may either resend I1 within the retry bounds or abandon the HIP exchange.
2. The system must extract the keying material from the ENCRYPTED parameter. This key is concatenated with that sent in the I2 packet to form the HIP data keys.

6.8. Sending UPDATE Packets

A host sends an UPDATE packet when it wants to update some information related to a HIP association. DEX UPDATE handling is the similar in DEX as in BEX. The key difference is NO HIP_SIGNATURE.

6.9. Handling State Loss

In the case of system crash and unanticipated state loss, the system SHOULD delete the corresponding HIP state, including the keying material. That is, the state SHOULD NOT be stored on stable storage. If the implementation does drop the state (as RECOMMENDED), it MUST also drop the peer's R1 generation counter value, unless a local policy explicitly defines that the value of that particular host is stored. An implementation MUST NOT store R1 generation counters by default, but storing R1 generation counter values, if done, MUST be configured by explicit HITs.

7. HIP Policies

There are a number of variables that will influence the HIP exchanges that each host must support. All HIP implementations MUST support more than one simultaneous HI, at least one of which SHOULD be reserved for anonymous usage. Although anonymous HIs will be rarely used as Responders' HIs, they will be common for Initiators. Support for more than two HIs is RECOMMENDED.

Many Initiators would want to use a different HI for different Responders. The implementations SHOULD provide for an ACL of Initiator's HIT to Responder's HIT. This ACL SHOULD also include preferred transform and local lifetimes.

The value of K used in the HIP R1 packet can also vary by policy. K should never be greater than 20, but for trusted partners it could be as low as 0.

Responders would need a similar ACL, representing which hosts they accept HIP exchanges, and the preferred transform and local lifetimes. Wildcarding SHOULD be supported for this ACL also.

8. Security Considerations

HIP is designed to provide secure authentication of hosts. HIP also attempts to limit the exposure of the host to various denial-of-service and man-in-the-middle (MitM) attacks. In so doing, HIP itself is subject to its own DoS and MitM attacks that potentially could be more damaging to a host's ability to conduct business as usual.

HIP DEX replaces the SIGMA authenticated Diffie-Hellman key exchange of BEX with a random generated key exchange encrypted by a Diffie-Hellman derived key. Both the Initiator and Responder contribute to this key.

The strength of the key is based on the quality of the secrets generated the Initiator and Responder. Since the Initiator is commonly a sensor there is a natural concern about the quality of its random number generator.

DEX lacks Perfect Forward Secrecy (PFS). If the Initiator's HI is compromised, ALL HIP connections protected with that HI are compromised.

The puzzle mechanism using CMAC may need further study that it does present the desired level of difficulty.

The DEX HIT extraction MAY present new attack opportunities; further study is needed.

The R1 packet is unprotected and offers an attacker new resource attacks against the Initiator. This is mitigated by the Initiator only processing a received R1 when it has sent an I1. This is another DoS attack, but for battery powered Initiators, it could be a concern.

9. IANA Considerations

IANA has reserved protocol number 139 for the Host Identity Protocol.

The following HIT suites are defined for DEX HIT generation.

Index	Hash function	Signature algorithm family	Description
5	LTRUNC	ECDH	ECDH HI truncated to 96 bits

Table 5: HIT Suites

10. Acknowledgments

The drive to put HIP on a cryptographic 'Diet' came out of a number of discussions with sensor vendors at IEEE 802.15 meetings. David McGrew was very

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2463] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

- [RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.
- [RFC4843-bis] Nikander, P., Laganier, J., and F. Dupont, "STUB: An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", draft-laganier-rfc4843-bis-00 (work in progress), February 2010.
- [RFC5201-bis] Moskowitz, R., Jokela, P., Henderson, T., and T. Heer, "Host Identity Protocol", draft-moskowitz-hip-rfc5201-bis-01 (work in progress), July 2010.
- [RFC5202] Jokela, P., Moskowitz, R., and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, April 2008.
- [fundamental-ecc] McGrew, D. and K. Igoe, "Fundamental Elliptic Curve Cryptography Algorithms", draft-mcgrew-fundamental-ecc-02 (work in progress), February 2010.

11.2. Informative References

- [AUR03] Aura, T., Nagarajan, A., and A. Gurtov, "Analysis of the HIP Base Exchange Protocol", in Proceedings of 10th Australasian Conference on Information Security and Privacy, July 2003.
- [CRO03] Crosby, SA. and DS. Wallach, "Denial of Service via Algorithmic Complexity Attacks", in Proceedings of Usenix Security Symposium 2003, Washington, DC., August 2003.
- [FIPS.197.2001] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <<http://csrc.nist.gov/>>

publications/fips/fips197/fips-197.pdf>.

- [IEEE.802-11.2007] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, June 2007, <<http://standards.ieee.org/getieee802/download/802.11-2007.pdf>>.
- [IEEE.802-15-4.2006] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Standard 802.15.4, September 2006, <<http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>>.
- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.
- [RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.
- [rfc4423-bis] Moskowitz, R., "Host Identity Protocol Architecture", draft-moskowitz-hip-rfc4423-bis-01 (work in progress), June 2010.

Appendix A. Using Responder Puzzles

As mentioned in Section 4.1.1, the Responder may delay state creation and still reject most spoofed I2s by using a number of pre-calculated RIs and a local selection function. This appendix defines one possible implementation in detail. The purpose of this appendix is to give the implementors an idea on how to implement the mechanism. If the implementation is based on this appendix, it MAY contain some local modification that makes an attacker's task harder.

The Responder creates a secret value *S*, that it regenerates

periodically. The Responder needs to remember the two latest values of S. Each time the S is regenerated, the R1 generation counter value is incremented by one and the Responder generates an R1 packet.

When the Initiator sends the I1 packet for initializing a connection, the Responder gets the HIT and IP address from the packet, and generates an I value for the puzzle.

```
I value calculation:
I = Ltrunc( CMAC ( S, HIT-I | HIT-R | IP-I | IP-R ), n)
where n = CMAC-len
```

From an incoming I2 packet, the Responder gets the required information to validate the puzzle: HITs, IP addresses, and the information of the used S value from the R1_COUNTER. Using these values, the Responder can regenerate the I, and verify it against the I received in the I2 packet. If the I values match, it can verify the solution using I, J, and difficulty K. If the I values do not match, the I2 is dropped.

```
puzzle_check:
V := Ltrunc( CMAC( I2.I | I2.I, I2.hit_i | I2.hit_r | I2.J ), K )
if V != 0, drop the packet
```

If the puzzle solution is correct, the I and J values are stored for later use. They are used as input material when keying material is generated.

Keeping state about failed puzzle solutions depends on the implementation. Although it is possible for the Responder not to keep any state information, it still may do so to protect itself against certain attacks (see Section 4.1.1).

Appendix B. Generating a Public Key Encoding from an HI

The following pseudo-code illustrates the process to generate a public key encoding from an HI for ECDH.

Author's Address

Robert Moskowitz
ICSA labs, An Independent Division of Verizon Business
1000 Bent Creek Blvd, Suite 200
Mechanicsburg, PA
USA

EMail: robert.moskowitz@icsalabs.com

Core
Internet-Draft
Intended status: Informational
Expires: April 22, 2011

C. O'Flynn
Atmel Corporation
B. Sarikaya
Huawei USA
Y. Ohba
Toshiba
Z. Cao
China Mobile
R. Cragie
Pacific Gas and Electric
October 19, 2010

Security Bootstrapping of Resource-Constrained Devices
draft-oflynn-core-bootstrapping-02

Abstract

The Internet of Things is marching its way towards completion. Nodes can use standards from the 6LoWPAN and ROLL WG to achieve IP connectivity. IEEE Standards ensure connectivity at lower layers for resource-constrained devices. Yet a central problem remains at a more basic layer without a suitable answer: how to initially configure the network. Without configuration the network never advances beyond a large box of nodes. Current solutions tend to be specific to a certain vendor, node type, or application.

This document outlines exactly what problems are faced in solving this problem. General problems faced in any low-power wireless network are outlined first; followed by how these apply to bootstrapping. A selection of currently proposed techniques is presented. From these a more generic approach is presented, which can solve the problem for a wide range of situations.

An emphasis is on performing this bootstrapping in a secure manner. This document does not cover operation of the network securely. This document does provide the basis for allowing the network to operate securely however, by providing standard methods for key exchanges and authentication.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-

Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on April 22, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	5
1.1. What is Bootstrapping?	5
1.2. Why IETF?	5
2. Bootstrapping Architecture	6
2.1. Areas of Booststrapping	6
2.2. Architecture	7
3. Communications Channel	8
3.1. Supported Communication Channels	8
4. Bootstrap Security Method	8
4.1. None	9
4.2. Asymmetric with User Authentication, Followed by Symmetric	9
4.3. Asymmetric with Certificate Authority, Followed by Symmetric	9
4.4. Cryptographically Generated Address Based Address Ownership Verification	9
5. Bootstrap Protocols	9
5.1. EAP Authentication Framework	10
5.2. PANA	11
5.3. HIP-DEX	13
5.4. 802.1X	13
6. Security Considerations	14
7. IANA Considerations	14
8. Acknowledgements	15
9. References	15
9.1. Normative References	15
9.2. Informative References	16
Appendix A. Examples of Node Configuration	18
A.1. Smart Energy	18
A.1.1. Initial Meter Installation	18
A.1.2. Home Expansions	18
A.2. Consumer Products	18
A.2.1. Connecting DVD Remote to DVD Player	18
A.2.2. Adding a TV to a network with a DVD player and remote	18
A.2.3. Providing GPS Location Data	18
A.3. Commercial Building Automation	19
A.3.1. Light Installation	19
Appendix B. Example Exchanges	19
B.1. Smart Energy: Meter Manufacture	19
B.2. Smart Energy: Meter Installation	19
B.3. Smart Energy: Home Expansion	19
B.4. Consumer: Connecting DVD Remote to DVD Player	19
B.5. Consumer: Adding a TV to a network with a DVD player and remote	21
B.6. Consumer: Providing GPS Location Data	23

B.7. Commercial: Building Automation	23
Authors' Addresses	23

1. Introduction

Familiarity with constrained network types is assumed here. Documents produced in the 6LoWPAN, ROLL, and CoRE Working Groups (WGs) would be a useful reference for the reader. In particular RFC 4919 [RFC4919] from 6LoWPAN, RFC 5548 [RFC5548] and RFC 5673 [RFC5673] from ROLL, CoAP [I-D.ietf-core-coap] from CoRE, and a paper by Romer and Mattern [ROMER04]. Familiarity with application specific examples such as Zigbee or Smart Energy groups is assumed.

A summary of those will be presented, as far as network requirements are concerned. The general network requirements will be further concentrated into requirements surrounding only the bootstrapping issues.

A number of solutions which are currently in use will be presented. Requirements on each solution will be stated to enable their use as a security bootstrapping protocol.

1.1. What is Bootstrapping?

Node configuration is known as bootstrapping in this document. Bootstrapping is any processing required before the network can operate. Typically this will require a number of settings to be transferred between nodes at all layers. This could include anything from link-layer information (i.e., wireless channels, link-layer encryption keys) to application-layer information (i.e., network names, application encryption keys).

Bootstrapping is complete when settings have been securely transferred prior to normal operation in the network.

1.2. Why IETF?

The bootstrapping problem is not specific to any MAC or PHY. This problem exists across any two nodes which have no previous knowledge of each other. In particular, this problem is complicated when the nodes are resource-constrained and may not have an advanced user interface. The IETF is instrumental in defining standards which will be used by The Internet of Things. Ensuring these standards can be used across nodes and networks requires some form of bootstrapping which any node can use.

Existing standards will be used as much as possible in this document. The method proposed here should work across many different underlying layers. It could be used to allow two nodes on the same physical network to join at the physical layer, or allow two nodes on an incompatible physical network to join at the IPv6 layer.

2. Bootstrapping Architecture

2.1. Areas of Bootstrapping

In order to provide a flexible architecture, the bootstrapping method is split into five distinct areas and two distinct phases. The five areas are a 'user interface', 'bootstrap profile', 'security method', 'bootstrap protocol', and the 'communications channel'.

The phases are provisioning phase and bootstrapping phase. In the provisioning phase, statically configured parameters (e.g., certificates) needed for the bootstrapping phase is provisioned. In the bootstrapping phase, dynamically configured information is set up using the statically configured information provided in the provisioning phase.

The user interface provides both user input and user output. Simple nodes may only have a push-button and LED, more complex nodes may have a graphical display and keyboard. The user interface provides interaction between the user and bootstrapping methods. The user interface would be used during bootstrapping as an OOB channel. It may also be used to specify bootstrapping policies.

The user interface provides the interaction between the user and the bootstrap protocol. The user interface will vary depending on the capabilities of the node. Examples might include a push-button and LED on simple nodes, to full-blown graphical user interfaces. Note that a 'bootstrapping tool' used to initially deploy a network is just a special user interface. This allows a very uniform protocol in deployment and use of networks.

User interface is out-of-scope and will not be further discussed.

Two nodes communicate through some channel. For our purposes this is split into the 'control channel' and 'data channel'. The control channel is used for the bootstrap protocol, and the data channel is used during normal network operation. A node may support multiple control or data channels. When the control and data channels are the same, the bootstrapping is done In Band (IB). When the control and data channels are different, the bootstrapping is performed Out Of Band (OOB). An 802.15.4 network for instance would use an 802.15.4 control channel for IB bootstrapping, but a control channel of perhaps IrDA or USB for OOB bootstrapping.

The 'bootstrap profile', i.e. statically configured parameters during the provisioning phase, defines what information should be exchanged during the process. A single node may run the protocol multiple times with different profiles. If the user wishes to associate a new

lightswitch, the protocol is first run with the '802.15.4 Wireless Profile', through which it learns the channel and PAN-ID. The node then runs a 'Security Exchange Profile' to learn the needed encryption keys. Finally it runs a 'Lightswitch Association Profile' through which it learns which light to associate with.

The 'security method' defines supported security methods for bootstrapping. The supported security methods will depend on the control channel and bootstrap profile. In one node if the control channel is secure, then a simple clear-text security method is supported. For example when a physical connection between two nodes is used, the control channel is considered secure. However when the control channel is not secure, this clear-text security method is not supported. The 'bootstrap profile' additionally defines allowed security methods. Higher security nodes may outlaw ever performing a clear-text exchange, even if the control channel is deemed secure.

The 'bootstrap protocol' defines the actual messages exchanged during bootstrapping. The messages are used to transfer between nodes data, node information, and network state. The selected security method runs on top of the control channel, such as EAP-GPSK etc.

2.2. Architecture

Security bootstrapping architecture is structured in a hierarchy of nodes going from the least resource constraint to the most resource constraint. At the top there is a root node. The root node is called Coordinator or Trust Center in Zigbee and 6LowWAN Border Router (6LBR) in 6LoWPAN ND.

At the next level there are interior Routers. Routers are able to run a routing protocol between other routers and the root. Router are called 6LowWAN Routers (6BR) in 6LoWPAN ND.

At the lowest level there are the nodes. The nodes do not run a routing protocol. They can connect to the nearest router over a single radio link. The nodes are called End Device in Zigbee and host in in 6LoWPAN ND.

Routers first join the network as a node and go through security bootstrapping operations in order to create a Master Session Key (MSK). Next routers execute routing protocol, e.g. [I-D.ietf-roll-rpl] specific steps to create session keys with their neighbors and to establish upstream and downstream next hop parents.

At each node hierarchy level described above, there are lower-layer and higher-layer protocols to bootstrap their ciphering keys, where the lower-layer refers to layers below IP layer including IEEE

802.15.4 MAC layer and LoWPAN adaptation layer and the higher-layer refers to IP layer and the above. In general, required bootstrapping procedures depend on the bootstrapping protocols to use. Section 5 describes the bootstrapping procedures where EAP (Extensible Authentication Protocol) [RFC3748] and other protocols are used as the bootstrapping protocols.

3. Communications Channel

The communications channel is the method used between two nodes to communicate. There are two main communication channels: the 'control' and 'data' channels. The control channel is used during bootstrapping, and the data channel is used during network operation.

3.1. Supported Communication Channels

There is no limit on what communications channels are supported. The following gives an example of several supported channels:

- o IEEE 802.15.4
- o Power-Line Communications
- o IrDA
- o RFID
- o Some simple physical link
- o Cellular
- o Ethernet
- o IPv6
- o Wi-Fi

Depending on the node's function, it may use different channels as the data or control channel. Nodes may have multiple data and/or control channels as well.

4. Bootstrap Security Method

The bootstrap security method defines allowable security methods. A node may choose to support or use a subset of these methods. This is NOT the security architecture used for the application, but only the

security used during bootstrapping. Typically some high-security method is used to generate a shared secret, which then switches to simpler symmetric encryption to secure the actual bootstrapping channel. The techniques negotiated should take advantage of hardware resources available, such as hardware encryption accelerators on an end node.

4.1. None

This is the simplest security method. No encryption or authentication is provided, messages are exchanged completely in clear-text. It is assumed some other layer provides security, such as a physical connection between devices.

4.2. Asymmetric with User Authentication, Followed by Symmetric

A Diffie-Hellman style key exchange is used to generate a shared secret. The authentication will be provided by the user, by confirming cryptographic signatures between two devices. With the shared secret generated through the DH, some symmetric encryption is used to secure the actual bootstrapping channel.

4.3. Asymmetric with Certificate Authority, Followed by Symmetric

Public key exchanges are used (aka: DH again), but with a Certificate Authority. Once a shared secret exists, symmetric encryption is used to secure the actual bootstrapping channel.

4.4. Cryptographically Generated Address Based Address Ownership Verification

A node may generate the global unique address using different techniques other than the stateless address autoconfiguration. For example, Cryptographically Generated Addresses (CGA) [RFC3972] is a type of global unique address that can be used to verify the address ownership. When the node uses CGA, it MUST execute SeND protocol [RFC3971]. In a 6LOWPAN network, a modified 6LOWPAN ND Protocol [I-D.ietf-6lowpan-nd] must be executed between the node and the edge router.

5. Bootstrap Protocols

In this section we first present EAP authentication framework and then describe three different protocols.

5.1. EAP Authentication Framework

In EAP, there are three distinct entities: the client or EAP peer, the authenticator and the authentication or EAP server [RFC5247].

The EAP peer is the node that requires to be authenticated before being admitted to the network. The authentication server is the device authenticating the node for bootstrapping. The authenticator is the device that is admitting the node to the network and it resides in between the peer and authentication server.

EAP client and EAP server exchange EAP messages to execute the authentication algorithm, a.k.a. EAP method. The authenticator is responsible for forwarding EAP messages between the client and server. In 802.1X, EAP messages are carried in Layer 2 and in PANA in IP or Layer 3. EAP messages between the authenticator and authentication server are carried using AAA protocols (RADIUS or Diameter).

At the end of a successful EAP method execution a master session key (MSK) is generated at both the EAP peer and EAP server. Authenticator receives MSK from EAP server at the end of EAP method execution using key transport. MSK is used in deriving a session key between the node and the authenticator using a protocol called secure association protocol (SAP). Derivation of the session key terminates bootstrapping of a node.

Additional keying material derived between EAP client and server that is exported by the EAP method is called Extended Master Session Key (EMSK). EMSK is not used in session key derivation but it could be used for the needs of other applications in higher layer protocols.

In the architecture introduced in Section 2.2 the node or router is the peer and the root is the authenticator. When the supplicant and authenticator are one hop away the authenticator can be reached directly. However, this is rarely the case. In other cases the authenticator authenticates neighboring supplicants first. The router nodes that are authenticated become relay authenticators in the next phase and they relay authentication messages from the supplicants to the authenticator and vice versa. This continues until all nodes are authenticated.

EAP is a lock-step protocol, i.e. it executes in pairs of EAP-Request messages sent by the server and EAP-Response messages sent by the peer. At the end, the server indicates the status of authentication, usually by EAP-Success message which also carries the MSK. The first EAP-Request/Response pair is used for the server to request the identity and the peer to provide it. In the other pairs of EAP

exchanges EAP method is executed.

Several EAP methods have been standardized each for different purposes. To authenticate devices with certificates, EAP Transport Layer Security (TLS) v1.2 specified in [RFC5216] which supports certificate-based mutual authentication is used.

Smart Energy Profile 2.0 Application Protocol Specification [SE2.0] mandates each device to be factory programmed with a certificate. The certificate is bound to a unique network ID, e.g. the device's MAC address or EUI-64 address. During EAP-Identity exchange the EAP peer provides its EUI-64 address as an identity to EAP server. This enables the server to validate the device certificate.

5.2. PANA

PANA (Protocol for carrying Authentication for Network Access) [RFC5191] defines an EAP transport over UDP where a PANA Client (PaC) is an EAP peer and a PANA Authentication Agent (PAA) is an EAP authenticator. There are three bootstrapping scenarios using PANA.

1. Use of PANA for bootstrapping link-layer security.

In this case, PANA is used for network access authentication to bootstrap link-layer ciphering. Security for higher-layer (i.e., IP layer and above) protocols is bootstrapped from an IB or OOB mechanism other than PANA. For example, in a 6LoWPAN deployment PANA authentication can take place to bootstrap IEEE 802.15.4 MAC layer ciphering keys.

2. In ZigBee IP, IEEE 802.15.4 MAC layer ciphering keys used as session keys are derived from a group key so called a Network Key that is securely distributed to each joining node upon successful PANA authentication using AES key wrap over PANA [I-D.ohba-pana-keywrap] where the key encryption key is derived from the EAP MSK (Master Session Key) [RFC3748].

3. Use of PANA for bootstrapping higher-layer security.

In this scenario, PANA is used as an OOB mechanism for higher-layer authentication to bootstrap ciphering keys for one or more higher-layer protocols independently of network access authentication. The PAA may reside in a higher-layer network element such as an ANSI C12.22 authentication host [C1222] and a CoAP server, or an independent server dedicated for service authentication for multiple higher-layer protocols. When bootstrapping ANSI C12.22 security for which no IB key management mechanism is available, ANSI C12.22 ciphering keys are directly

derived from EAP key material established from PANA authentication. When bootstrapping CoAP security, a PSK (Pre-Shared Key) credential in the combined usage of DTLS (Datagram Transport Layer Security) [RFC4347] and PSK mode of TLS [RFC4279] is derived from EAP key material and DTLS ciphering keys are generated as a result of a successful DTLS handshake.

The ability to bootstrap multiple higher-layer protocols from a single execution of PANA authentication is important to save the computational resources for resource-constrained devices especially where public-key based authentication is used.

4. Use of PANA for bootstrapping both link-layer and higher-layer security.

This case is the combination of the other two cases, and the most optimized way for bootstrapping resource-constrained devices. This case is only applicable where both the network access authentication and the higher-layer authentication use the same authentication server with the same authentication credentials.

The second and third scenarios are generally referred to as Single Sign-On in section 4.2.2.2 of [NISTIR7628VOL1], where the root keys for higher-layer protocols can be derived from EAP EMSK (Extended Master Session Key) as an USRK (Usage-Specific Root Key) [RFC5295].

A PANA Relay Element (PRE) is needed to enable PANA messaging between a PANA Client (PaC) which is the node to be authenticated and a PANA Authentication Agent (PAA) which is the authenticator where the two nodes cannot reach each other by means of regular IP routing. This happens during authentication since only a link-local IPv6 address can be used prior to the completion of a successful authentication.

PRE which is one hop away from PaC receives PANA messages and relays the message contents (payload) by encapsulating it in a message parameter called Attribute Value Pair (AVP). PRE also needs to send header contents such as PaC's IP address and UDP port number in a different AVP. PRE has IP routing established with PAA which could be several hops away. PAA sends its reply messages in which the payload is encapsulated in an AVP. It also adds the AVP containing PaC's IP address and UDP port number. PRE sends creates a link local PDU using these AVPs and sends it to PaC.

The requirements for the use of PANA as a bootstrapping protocol can be stated as follows:

- o A new entity called PANA Relay Element needs to be added to the PANA architecture. Behaviour of PANA Relay Element needs to be

defined.

- o New AVPs needed for PANA Relay Element operation for properly relaying messages from the client to the authenticator and vice versa are required to be specified.
- o An extension to PANA to securely distribute keys from the PANA Authentication Agent to the PANA Client using AES Key Wrap with Padding algorithm needs to be defined. This is needed in order to use PANA for group key distribution.

5.3. HIP-DEX

[RFC4423] introduces the Host Identity Protocol (HIP) where the Host Identity (HI) is a Cryptographic key (RSA, DSA, or ECC). A 128-bit length Host Identity Tag (HIT) is derived from the HI (hashed) and functions as an IPv6 address (/128 prefix) for applications. A four-packet Peer-to-Peer Host Identity Protocol Base EXchange (HIP BEX) establishes a security association (SA, similar to IKE), indexed by the HITs, but independent of the IP address. So HIP can be considered as a shim layer between the transport(TCP/UDP) and IP, providing authentication, data confidentiality, mobility in one basket.

The HIP BEX involves many crypto primitives that are difficult to run on constrained nodes. HIP Diet Exchange (HIP DEX) [I-D.moskowitz-hip-rg-dex] is a way to make HIP lightweight. Basically, HIP DEX a variant of the HIP BEX specifically designed to use as few crypto primitives as possible yet still provide the same class of security features as HIP BEX.

In the architecture introduced in Section 2.2 the node and router could be the HIP end-points. Depending on who initiates the HIP Diet Exchange, the node or router could act as the HIP initiator and HIP responder respectively. And the initiator and responder can be multiple hops way from each other, as long as there is an IP connectivity between them.

An important requirement for the HIP-DEX to work in the architecture, the initiator should be able to get the IP address of the responder, either using DNS infrastructure or local configuration.

5.4. 802.1X

IEEE 802.1X defines how EAP packets can be transported over in Layer 2, i.e. Ethernet frames [802.1x] by encapsulating EAP packets into EAP Over Lan (EAPOL) frames between EAP peer, called supplicant and the authenticator. EAPOL can also be used in 802.11 wireless links.

To enable IEEE 802.15.4 devices to use EAP authentication, EAP packets encapsulated in EAPOL frames can be carried as payload in 802.15.4 data frames [802.15.4]. EAPOL is well defined and widely used and it lends itself to be easily carried in 802.15.4 data frames. For this, Frame Type subfield of the Frame Control Field of IEEE 802.15.4 MAC header needs to be set to a special value to indicate the type of the payload, i.e. 802.1X encapsulated EAP packets. EAPOL packets are encoded following common EAPOL PDU structure defined in [802.1x] into the data payload field of 802.15.4 data frames.

Authentication proceeds as follows: authenticator authenticates the supplicants that are on the next hop first. This enables a secure link between the authenticator and these first-hop nodes. First-hop nodes or router become Relay Authenticators in the next phase of authentication. Relay authenticators tunnel EAPOL frames to the authenticator in the secure link established. This way all the supplicants are gradually authenticated.

The requirements for the use of 802.1X defined EAPOL as a bootstrapping protocol can be stated as follows:

- o A special value in the Frame Type subfield of the Frame Control Field of IEEE 802.15.4 MAC header to indicate the type of the payload,
- o Group addresses for 802.15.4 corresponding to EAPOL Group Address Assignments defined in Table 11.1 of [802.1x], especially to be used in EAPOL-Start packet.
- o Which MAC frames of beacon, data, acknowledgment and MAC command as defined in [802.15.4] with what security levels are mapped to controlled port, which MAC frames with what security levels are mapped to uncontrolled port and which MAC frames are never mapped to any of controlled/uncontrolled port (i.e., the payload of those frames are used by the MAC-layer itself and never used by upper layers).

6. Security Considerations

TBD.

7. IANA Considerations

This memo includes no request to IANA.

8. Acknowledgements

Thanks to Zach Shelby for editing, comments, and overall assistance.

9. References

9.1. Normative References

- [802.15.4] IEEE Std 802.15.4-2006, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)", September 2006.
- [802.1x] IEEE Std 802.1X-2010, "IEEE 802.1X Port-Based Network Access Control", February 2010.
- [RF4CE] ZigBee Alliance, "Zigbee RF4CE Specification Version 1.00", March 2009.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.
- [RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.
- [RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.
- [RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.
- [RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.
- [ROMER04] Romer, K. and F. Mattern, "The design space of wireless

sensor networks", IEEE Wireless Communications, vol. 11, no. 6, pp. 54-61, December 2004.

- [SE2.0] ZigBee Alliance, "Smart Energy Profile 2.0 Technical Requirements Document", April 2010.

9.2. Informative References

- [C1222] American National Standard, "Protocol Specification For Interfacing to Data Communication Networks", ANSI C12.22-2008, 2008.
- [I-D.ietf-6lowpan-nd]
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low-power and Lossy Networks", draft-ietf-6lowpan-nd-13 (work in progress), September 2010.
- [I-D.ietf-core-coap]
Shelby, Z., Frank, B., and D. Sturek, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-02 (work in progress), September 2010.
- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., Brandt, A., Clausen, T., Hui, J., Kelsey, R., Levis, P., Networks, D., Struik, R., and J. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-ietf-roll-rpl-12 (work in progress), October 2010.
- [I-D.moskowitz-hip-rg-dex]
Moskowitz, R., "HIP Diet EXchange (DEX)", draft-moskowitz-hip-rg-dex-02 (work in progress), July 2010.
- [I-D.narten-iana-considerations-rfc2434bis]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.
- [I-D.ohba-pana-keywrap]
Chakrabarti, S., Cragie, R., Duffy, P., Ohba, Y., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Extension for Key Wrap", draft-ohba-pana-keywrap-01 (work in progress), October 2010.

[NISTIR7628VOL1]

The Smart Grid Interoperability Panel - Cyber Security Working Group, "Guidelines for Smart Grid Cyber Security: Vol. 1, Smart Grid Cyber Security Strategy, Architecture, and High-Level Requirements", NISTIR 7628, vol. 1, 2010.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

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

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

[RFC5295] Salowey, J., Dondeti, L., Narayanan, V., and M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, August 2008.

[RFC5433] Clancy, T. and H. Tschofenig, "Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method", RFC 5433, February 2009.

Appendix A. Examples of Node Configuration

Before any detail on methods is explored, the following section will provide various examples this document could cover. Exact requirements will be brought forward in subsequent sections. For the reader's general understanding this section is placed to give an idea of an acceptable usage scenario.

A.1. Smart Energy

A.1.1. Initial Meter Installation

The meter is initially loaded with code and network keys through a physical interface at the factory. The meter is installed at a customers home, and configured by the installer through the backbone link (via GSM modem, etc). Both operations can be performed through methods defined herein.

A.1.2. Home Expansions

The user wishes to join a thermostat onto the network. They press a button on the thermostat, which enters join mode. They press a button on the smart meter, which allows nodes to join the network. The devices both have displays, so they display a certain number which the user verifies match on both devices. The thermostat has now securely joined the network.

A.2. Consumer Products

A.2.1. Connecting DVD Remote to DVD Player

The user pushes a join button on the DVD remote and DVD player. The devices find each other, and blink in unison to indicate to the user which two devices will join. The user presses the button to confirm this, and the two devices are now joined together.

A.2.2. Adding a TV to a network with a DVD player and remote

The user then presses the join button on the DVD player and TV. The devices again find each other and blink in unison, with the addition that the remote control also blinks to indicate it is present in the network.

A.2.3. Providing GPS Location Data

A user has a simple GPS receiver (that has no user interface) they wish to broadcast location data with. The user switches on their camera, and enters a PIN from the base of the GPS. The user can now

view GPS information such as satellite health from their camera. In addition photos are automatically tagged with location information.

A.3. Commercial Building Automation

A.3.1. Light Installation

The electrician installs the light fixture. Each light has a barcode printed on it. They use a handheld barcode scanner tool, which acts as the commissioning tool. When they scan a barcode with the tool, the tool asks the electrician to enter some additional information such as light fixture location. The tool securely registers the light fixture on the network, along with setting parameters inside the light fixture.

Appendix B. Example Exchanges

The following details how the protocol handles certain conditions and situations through examples. Note that each example is a more detailed description of the examples in Appendix A.

B.1. Smart Energy: Meter Manufacture

B.2. Smart Energy: Meter Installation

B.3. Smart Energy: Home Expansion

B.4. Consumer: Connecting DVD Remote to DVD Player

Supported User Interface Profiles

Profile	DVD Player	Remote Control
none	Y	Y
simple	Y	Y
numerical	Y	N
alphanumeric	Y	N
Graphical	Y	N

Supported Bootstrap Transport Layers

Layer	DVD Player	Remote Control
Physical	Y	Y
802.15.4	Y	Y
IrDA	Y	N

Supported Security Methods

Method	DVD Player	Remote Control
None	Y	Y
EAP	Y	N
Asymmetric, User	Y	Y
Asymmetric, CA	Y	N

The DVD player and remote control have a number of ways in which they could be joined together. The remote control does not have any unique identifier printed on it, thus no pre-shared key can be identified. This leaves either an unsecure joining method, or some asymmetric security method.

The remote control has a button on it for 'join', as does the DVD player. The user pushes the button on the DVD player, and then pushes the button on the remote control. Based on the UI profile, this causes the following to occur:

- o DVD Player scans for existing network in advertise mode. Finding none, it starts a new network and that network enters advertise mode.
- o The DVD remote scans for a network, and then finds the DVD player's network.
- o The devices generate a shared secret (ie: Diffie-Hellman), and both blink their LED in a unique pattern based on this shared secret.
- o The user confirms both devices are blinking the same pattern, as both LEDs are blinking in unison.
- o The DVD player displays 'JOIN OK' on it's LCD panel.

B.5. Consumer: Adding a TV to a network with a DVD player and remote

This network will have three devices: a TV, a DVD Player, and a Remote Control. The user will run the bootstrap protocol between the TV and Remote Control in this example, although it could also be run between the TV and DVD player.

Supported User Interface Profiles

Profile	TV	Remote Control
none	Y	Y
simple	Y	Y
numerical	Y	N
alphanumeric	Y	N
Graphical	Y	N

Supported Bootstrap Transport Layers

Layer	TV	Remote Control
Physical	Y	Y
802.15.4	Y	Y
IrDA	Y	N

Supported Security Methods

Method	TV	Remote Control
None	Y	Y
EAP-GPSK	Y	N
Asymmetric, User	Y	Y
Asymmetric, CA	Y	N

The TV and remote control have a number of ways in which they could be joined together. The remote control does not have any unique identifier printed on it, thus no pre-shared key can be identified. This leaves either an unsecure joining method, or some asymmetric security method.

The remote control has a button on it for 'join', as does the TV. In this example two sequence will be considered: where the TV button is

pressed first, and where the remote control button is pressed first.

If the TV join button is pressed first:

- o TV scans for existing networks in advertise mode. Finding none, it starts a new network and that network enters advertise mode.
- o The remote scans for a network, and then finds the TV's network.
- o The remote informs the TV it is on an existing network, and thus will require the TV to join this network.
- o The devices generate a shared secret, and both blink their LED in a unique pattern.
- o The DVD player in addition blinks, so the user is informed that if they confirm the join action the resulting network will have all three devices in it.
- o The user confirms both devices are blinking the same pattern, as both LEDs are blinking in unison.
- o The TV displays 'JOIN OK' onscreen, along with any information about the network it just joined.

If the remote control join button is pressed first:

- o Remote control scans for existing networks in advertise mode. Finding none, it advertises it's network.
- o The TV scans for a network, and then finds the remote control's network.
- o The devices generate a shared secret, and both blink their LED in a unique pattern.
- o The DVD player in addition blinks, so the user is informed that if they confirm the join action the resulting network will have all three devices in it.
- o The user confirms both devices are blinking the same pattern, as both LEDs are blinking in unison.
- o The TV displays 'JOIN OK' onscreen, along with any information about the network it just joined.

B.6. Consumer: Providing GPS Location Data

B.7. Commercial: Building Automation

Authors' Addresses

Colin Patrick O'Flynn
Atmel Corporation
Colorado Springs, Colorado
USA

Phone:
Email: colin.oflynn@atmel.com

Behcet Sarikaya
Huawei USA
1700 Alma Dr. Suite 500
Plano, TX 75075

Email: sarikaya@ieee.org

Yoshihiro Ohba
Toshiba
Tokyo, Japan

Email: yoshihiro.ohba@toshiba.co.jp

Zhen Cao
China Mobile
Beijing, China

Email: caozhen@chinamobile.com

Robert Cragie
Pacific Gas and Electric
89 Greenfield Crescent
Wakefield, UK WF4 4WA

Email: robert.cragie@gridmerge.com

6lowpan Working Group
Internet-Draft
Expires: April 29, 2011

Y. Qiu
J. Zhou
F. Bao
Institute for Infocomm Research
October 26, 2010

Lightweight Key Establishment and Management Protocol in Dynamic Sensor
Networks (KEMP)
draft-qi-6lowpan-secure-router-01

Abstract

When a sensor node roams within a very large and distributed wireless sensor network, which consists of numerous sensor nodes, its routing path and neighborhood keep changing. In order to provide a high level of security in this environment, the moving sensor node needs to be authenticated to new neighboring nodes as well as to establish a key for secure communication. The document proposes an efficient and scalable protocol to establish and update the secure key in a dynamic wireless sensor network environment. The protocol guarantees that two sensor nodes share at least one key with probability 1 (100%) with less memory and energy cost, while not causing considerable communication overhead.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 29, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

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

Table of Contents

1. Introduction	3
2. Network Assumptions	5
3. Shared-Key Discovery	6
4. Dynamic Authentication and Key Establishment Protocol	7
4.1. Basic Protocol	7
4.2. Key Management	8
4.3. Distribution Mode	10
5. Security Consideration	12
6. IANA Consideration	14
7. Conclusions	15
8. Normative References	16
Authors' Addresses	17

1. Introduction

The demand of wireless sensor networks (WSNs) is growing exponentially. It has turned out that the sensor networks can be widely applied in the areas of healthcare, environment monitoring, and the military. One of the surveys on WSNs points out that, in the near future, wireless sensor networks will be an integral part of our lives, more so than the present-day personal computer [1].

A sensor node has low capability in terms of power, computation, storage and communication. A wireless sensor network is composed of a large number of wireless sensor nodes and multi-hop communication is desired in WSNs. As a result, security in wireless sensor networks has six challenges to overcome: (a) the wireless nature of communication, (b) resource limitations of sensor nodes, (c) very large and dense WSNs, (d) lack of fixed infrastructure, (e) unknown network topology prior to deployment, (f) high risk of physical attacks on unattended sensors [2][3].

The capabilities in term of Scalability, Mobility/Dynamicity Network, Latency, etc. are also listed in the RFC documents, i.e. Routing Requirements for Urban Low-Power and Lossy Networks (RFC 5548)[6], Routing Requirements for Urban Low-Power and Lossy Networks (RFC 5673)[7], Home Automation Routing Requirements in Low-Power and Lossy Networks (RFC 5826)[8], and Building Automation Routing Requirements in Low-Power and Lossy Networks (RFC 5867)[9].

RFC 5548 required local network dynamics SHOULD NOT impact the entire network to be reorganized or re-reconfigured; a viable routing security approach SHOULD be sufficiently lightweight that it may be implemented across all nodes in a U-LLN; the U-LLN MUST deny any node that has not been authenticated to the U-LLN and authorized to participate to the routing decision process.

RFC 5673 addressed the handover speed; a compromised field device does not destroy the security of the whole network; because nodes are usually expected to be capable of routing, the end-node security requirements are usually a superset of the router requirements.

RFC 5826 needed a node MUST authenticate itself to a trusted node that is already associated with the LLN before the former can take part in self-configuration or self-organization. A node that has already authenticated and associated with the LLN MUST deny, to the maximum extent possible, the allocation of resources to any unauthenticated peer. The routing protocol(s) MUST deny service to any node that has not clearly established trust with the HC-LLN.

RFC 5867 listed the possible security keys below: a) a key obtained

from a trust center already operable on the LLN; b) a pre-shared static key as defined by the general contractor or its designee; or c) a well-known default static key.

With the aforementioned limitations of the existing solutions in mind, we now propose a secure protocol in dynamic WSN, addressing all of the following issues:

- o A moving sensor node needs to change its attached routers (or cluster heads) frequently.
- o A router (or cluster head) needs to ensure a joining node is not a malicious sensor.
- o A moving node needs to establish a secure tunnel with the new router (or cluster head).
- o The energy consumption for establishing the secure tunnel must be minimal.

One of the important novel features of the proposed protocol is that the router or cluster head is employed as sub-base-stations to execute key establishment. This way, the total dependency on the base station for key establishment can be avoided. Also, this approach reduces the hops between two communicating ends and hence results in reduction of the communication cost.

2. Network Assumptions

In this document, we consider a scenario in which a sensor node roams within a very large and distributed WSN, consisting of a large number of sensor nodes. It is a typical scenario that is widely adopted in hospital environments as the patients or doctors equipped with sensors roam across each department in the hospital. A patient who carries the sensor nodes can move freely within the range of a hospital. When a wireless sensor node is moving, its routing path and neighborhood keep changing. The moving node needs to be authenticated to the new neighbors and to establish a key for secure communication.

This scenario reflects the problems described in Section 1: (a) composition by a large number of sensor nodes; (b) communication based on wireless multi-hop mechanism; (c) no fixed infrastructure; (d) the possible location change of sensor node (patient). Therefore, the challenges of this network assumption are how to establish a secure channel with these routers.

3. Shared-Key Discovery

In the WSN environment, as data transmission consumes much more energy than computation, the probabilistic solution is widely accepted in order to reduce the storage and communication overhead during key establishment.

So far in the literature, numerous random key pre-distribution schemes have been proposed. For example, in Chan et al.'s scheme[4], each sensor node stores a random set of Np dedicated pair-wise keys to achieve the probability p that two nodes share a key. At the key setup phase, each node ID is matched with Np other randomly selected node IDs with probability p . A distinct pair-wise key is generated for each ID pair, and is stored in both nodes' key-chain along with the ID of the other party. During the shared-key discovery phase, each node broadcasts its ID so that neighboring nodes can tell if they share a common pair-wise key. Note that Chan et al.'s scheme reduces the storage overhead by sacrificing key connectivity, but it still provides perfect key resilience.

In this protocol, it is assumed that a sensor node (carried by a patient) can move within a special range (e.g. hospital). As each sensor's memory is severely constrained, each sensor may only store a small set of keys randomly selected from a key pool at the deployment. Two nodes may use any existing key discovery protocol (e.g., the solution proposed in [4]) to find a common key from their own sets. If the common key is not found, the key establishment scheme will be initiated. The reason why binding a general pre-shared key discovery phase to the protocol is to reduce the energy cost as much as possible.

4. Dynamic Authentication and Key Establishment Protocol

4.1. Basic Protocol

Due to the limited storage of sensor nodes, the pre-shared key-pair is not always available between the roaming node and its new neighbors in the circumstance of a dynamic node roaming within large WSNs (e.g., in hospitals and nuclear power plants). Therefore it requires an efficient and scalable protocol to establish and update the keys among nodes for secure communications.

Figure 1 shows the basic architecture and message flow of our protocol for authentication and key establishment in dynamic WSNs. When a dynamic sensor node moves to a new area and wants to attach to a router or a cluster head in this area, it first sends a request message to the base station (refer to Figure 1).

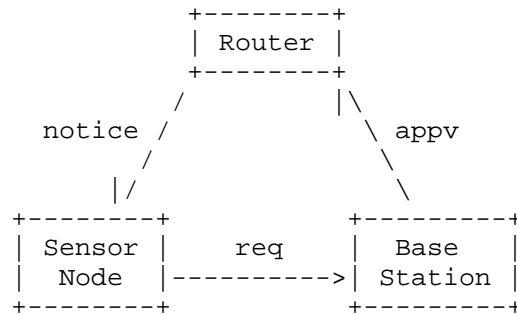


Figure 1. The basic architecture and message flow of KEMP protocol

$$\text{req} = \{\text{Src} = \text{SN}, \text{Dst} = \text{BS}, \text{RT} \parallel \text{R0} \parallel \text{MAC}(\text{K_BN}, \text{SN} \parallel \text{RT} \parallel \text{R0})\} \quad (1)$$

where Src and Dst denote the source and destination address of a message respectively. SN, BS and RT are identifiers for sensor node, base station and router, respectively. R0 denotes a random number generated by the sensor node. MAC indicates the message authentication code algorithm with a key and K_BN is the shared secret key between the base station and the sensor node.

After receiving the req message, the base station will check its revocation list whether the sensor node has been revoked. If the sensor node is acceptable, then the base station verifies the MAC message. If the result is positive, the base station will generate a session key K_NR for the roaming sensor node and the router (or cluster head).

$$K_{NR} = H(K_{BN}, SN || R0 || R1) \quad (2)$$

where H is a keyed one-way hash function, and $R1$ is the random number selected by the base station. The base station then sends an approval message $appv$ with the session key to the router:

$$appv = \{Src=BS, Dst=RT, E(K_{BR}, SN || R0 || R1 || K_{NR})\} \quad (3)$$

where E is an encryption algorithm, and K_{BR} is the shared secret key between the base station and the router.

After receiving the $appv$ message, the router decrypts the payload and extracts the session key K_{NR} , and then sends a notice to the sensor node.

$$notice = \{Src=RT, Dst=SN, R0 || R1 || MAC(K_{NR}, RT || SN || R0 || R1)\} \quad (4)$$

Upon getting the notice message, the sensor node extracts the random numbers $R0$ and $R1$. After checking if the received random number $R0$ is equal to the original $R0$, the sensor node recalculates the session key $K_{NR} = H(K_{BN}, SN || R0 || R1)$ and then verifies the MAC value. If the result is positive, the sensor node will use the session key for the communication with this router afterwards. In practice, the router could be any sensor node that the dynamic sensor node wants to connect to.

4.2. Key Management

In order to manage the keys, every sensor node maintains a table, called "Key Cache". Table 1 shows the structure of the Key Cache.

Table 1. The structure of Key Cache

Key Cache in Sensor Node N		
Correspondence Node ID	Key	Key Lifetime
BS	K_{BN}	T_{BN}
Node _i	K_{Ni}	T_{Ni}
...
Node _j	K_{Nj}	T_{Nj}
PreSharedKey _x	K_x	T_x
...
PreSharedKey _y	K_y	T_y

When a sensor node, say node N , wants to connect to other sensor

node, say node R, it executes the following procedure:

- (1) Checks first if there is an existing key pair between them.
- (2) Otherwise, processes the subroutine of shared-key discovery to find a common key between node N and node R based on those "PreSharedKeys" in their key caches.
- (3) If there is still no common key between them, the sensor node allocates an entry in the key cache, and assigns Node ID as nodeR, Key Stuff as the random number R0 and Key Lifetime as 0, as shown in Table 2.

Table 2. The initial key entry.

Correspondence Node ID Key Key Lifetime			
Node_R R0 0			

- (4) Then the sensor node initiates the procedure of key establishment described in the above section. After receiving the notice message, and recalculating the session key KNR, the sensor node updates the entry's key stuff and key lifetime accordingly.
- (5) When the key lifetime is expired, the dynamic sensor node should re-initiate the procedure of key establishment described in the above section.
- (6) When the sensor node leaves the range of the connected router, the sensor node deletes the related entry from its cache table in order to save the storage. In case there is no space for adding a new entry, it may first delete the oldest key which has expired or will expire soon.

The base station also maintains a key table (Table 3) that includes the secret keys shared with all of the sensor nodes in the network.

Table 3. The structure of Key Table in basestation

Key Table in Base Station		
Node ID	Key	Key Lifetime
Node_i	K_Bi	T_Bi
...
Node_j	K_Bj	T_Bj

+-----+

If a node is compromised and revoked, its field of key lifetime would be marked as negative.

4.3. Distribution Mode

In WSNs, the more hops between two communicating ends exist, the poorer the traffic performance becomes and the more energy consumption is required. To overcome these problems, we introduce the distribution mode.

The major idea of distribution mode is to deploy the cluster heads as the sub-base-stations because a cluster head is more powerful than normal sensor nodes. The distribution mode includes the following steps:

- (1) Each cluster head manages to establish the shared key with its neighboring cluster heads after deployment. There are several ways to do this. One could embed those keys in advance if the topology is known at deployment, or use the basic protocol described in the above sections, via the base station. (As this is a one-time operation, the overheads may be acceptable.)
- (2) Each sensor node keeps two base station identifiers (IDs): one is a real base station ID; the other is a sub-base-station (the cluster head) ID. Initially, the ID of sub-base-station is a real base station.
- (3) After deployment, the first round for a mobile node to establish the shared key with the nearest cluster head uses the basic protocol, too.
- (4) When the mobile node moves, use the basic protocol to establish the shared key with the new cluster head, via the sub-base-station (old cluster head) rather than the real base station.
- (5) After successfully establishing the keys, the sensor node updates the ID of sub-base-station with the current cluster head.
- (6) For security reasons, each sensor node must reset its sub-base-station ID to the real base station at a specified interval (say a few hours or days, depending on the various applications) and re-establish keys with its near cluster heads via the real base station. If the base station does not receive any request from a sensor node, it considers the sensor node has been

compromised.

The distribution mode could provide an efficient and low energy-cost solution for the shared-key establishment. The basic protocol can provide the stronger protection since it can immediately block and revoke compromised nodes.

5. Security Consideration

In this proposed protocol, the session key K_{NR} between the sensor node and the router is generated by the base station and sensor node respectively, and the session key is directly sent to the router from the base station by an encrypted packet. Hence, the session key K_{NR} is never disclosed during transmission. The session key K_{NR} is only known by the related peers, i.e., the sensor node, the base station and the router.

Referring to equation (2), the session key K_{NR} is generated by a keyed hash function with the shared key K_{BN} between sensor node and base station as well as two random numbers, $R0$ and $R1$, which are generated by the sensor node and base station respectively. As both $R0$ and $R1$ are used only one time, there are not the same session keys K_{NR} . This property is useful to against the replication attacks.

Since the session key K_{NR} is generated by a keyed hash function with the secret key K_{BN} between the sensor node and the base station, the different sensor nodes will have different session keys. This feature is useful to protect sensor node privacy.

Even though an eavesdropper at the edge of the sensor node can monitor and capture the random numbers $R0$ and $R1$ as well as the identity of the sensor node, it is still not able to regenerate the session key K_{NR} due to lack of the secret key K_{BN} . Without a proper session key, the routers will not forward the packets to next nodes. This attribute could prevent camouflage and traffic attacks.

Due to the fact that no trusted connection is established between sensor node and new router before the connection between them, the proposed protocol employs a random number $R1$ issued by the base station. The sensor node needs to recalculate the K_{NR} first based on the $R1$ together with K_{BN} and $R0$. Then using the calculated session key K_{NR} to verify the received session key K_{NR} and the random number $R1$. If the result is positive, then the sensor node will trust that the router is authorized by the base station.

Besides the function of informing the sensor node that the new session key K_{NR} is ready to use in the router, the notice message also plays an important role to check if the sensor node's address is reachable. Without this reachability check, the sensor node may claim that it is at any location rather than its real location. It could launch redirecting attacks.

The path between the base station and the router is secure because the packet between them is encrypted with a pre-shared key K_{BR} .

The messages from the sensor node to the base station and from the router to the sensor node are authenticated by a keyed hash function. Before accepting the inward message and making further processing, the receivers must verify the authentication. Since the cost of a hash algorithm is very small, the base station and sensor node could avoid the attacks of denial of service.

In order to achieve high efficiency and low energy cost, the protocol deploys a distribution mode which uses the cluster headers as the sub-base-stations. Due to the capability of cluster header, it is not able to recognize any compromised sensor nodes in time; the protocol requires each sensor node to reset its sub-base-station ID to the real base station regularly, and to re-establish keys with its near cluster heads via the real base station. This step is also useful to avoid a sensor node binding a compromised cluster head for long time.

According to the above analysis, this proposed protocol, which is simple and easy to implement, can provide relatively strong protection for sensor node networks.

6. IANA Consideration

This version does not need new values to be assigned by IANA.

7. Conclusions

In this document, we have proposed an efficient and scalable protocol to establish and update the authentication key between any pair of sensor nodes in a dynamic wireless sensor network. Our protocol has the following features:

- o It is suitable for both static and dynamic WSNs. Any pair of nodes can establish a key for secure communication.
- o A roaming node only deals with its closest router for security. There is no need to change the rest of routing path to the base station.
- o The base station can manage a revocation list for lost or compromised roaming nodes.
- o The system is scalable and resilient against node compromise.
- o The protocol is efficient due to the small number and size of signalling messages.
- o The size of each signalling message is smaller than the IEEE 802.15.4 frame size so that it can avoid packet fragmentation and the overhead for reassembly.
- o The distribution mode can considerably reduce the latency.
- o Any pair of nodes can establish a key. The protocol guarantees that two sensor nodes share at least one key with probability 1 (100%).

Thanks to above features, the protocol can satisfy the requirements for IPv6 over Low power WPAN Routing [5] and could be the security solution deployed in Routing Requirements for Urban Low-Power and Lossy Networks (RFC 5548)[6], Routing Requirements for Urban Low-Power and Lossy Networks (RFC 5673)[7], Home Automation Routing Requirements in Low-Power and Lossy Networks (RFC 5826)[8], and Building Automation Routing Requirements in Low-Power and Lossy Networks (RFC 5867)[9].

After comparing with some of the popular and latest protocols used in WSNs, our protocol could save about 30% in communication energy, and has the higher probability (100%) of sharing a key between two sensor nodes with less memory cost than those pre-distribution schemes, without incurring in a considerable amount of communication.

8. Normative References

- [1] Akyildiz, I., Sankarasubramaniam, Y., and E. Cayirci, "Wireless sensor networks: a survey", *Comput. Netw* 38, 393-422, 2002.
- [2] Camtepe, S. and B. Yener,, "Key Distribution Mechanisms for Wireless Sensor Networks: a Survey", Technical Report TR-05-07; Department of Computer Science, Rensselaer Polytechnic Institute: Troy, NY, USA , Mar. 2005.
- [3] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.
- [4] Chan, H., Perrig, A., and D. Song, "Random key predistribution schemes for sensor networks", *IEEE Symposium on Research in Security and Privacy* Oakland, California, USA, May 2003.
- [5] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for 6LoWPAN Routing", Work in Progress, Aug. 2010.
- [6] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.
- [7] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.
- [8] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, April 2010.
- [9] Martocci, J., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, June 2010.

Authors' Addresses

Ying Qiu
Institute for Infocomm Research, Singapore
1 Fusionopolis Way
#21-01 Connexis (South Tower)
Singapore 138632

Phone: +65-6408 2053
Email: qiuying@i2r.a-star.edu.sg

Jianying Zhou
Institute for Infocomm Research, Singapore
1 Fusionopolis Way
#21-01 Connexis (South Tower)
Singapore 138632

Phone: +65-6408 2075
Email: jyzhou@i2r.a-star.edu.sg

Feng Bao
Institute for Infocomm Research, Singapore
1 Fusionopolis Way
#21-01 Connexis (South Tower)
Singapore 138632

Phone: +65-6408 2073
Email: baofeng@i2r.a-star.edu.sg

6LoWPAN
Internet-Draft
Intended status: Standards Track
Expires: December 8, 2010

P. Thubert
Cisco
June 6, 2010

6LoWPAN Backbone Router
draft-thubert-6lowpan-backbone-router-02

Abstract

Some LLN subnets are expected to scale up to the thousands of nodes and hundreds of routers. This paper proposes an IPv6 version of the Backbone Router concept that enables such a degree of scalability using a high speed network as a backbone to the subnet.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 8, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Overview	6
4. New types and formats	8
4.1. Binding Tracking Option	8
5. Backbone Router Operations	10
5.1. Backbone Link and Router	10
5.2. ND Proxy Operations	10
5.3. Claiming and defending	12
5.4. Conflict Resolution	12
5.5. Assessing an entry	13
6. Security Considerations	15
7. IANA Considerations	16
8. Acknowledgments	17
9. References	18
9.1. Normative References	18
9.2. Informative References	18
Author's Address	20

1. Introduction

The ISA100.11a standard has introduced the concept of a Backbone Router that would interconnect small LLNs over a high speed transit network and scale a single ISA100.11a network up to the thousands of nodes. In that model the LLNs and the backbone form a single subnet in which nodes can move freely without the need of renumbering, and the Backbone Router is a special kind of Border Router designed to manage the interaction between the LLNs and the backbone at layer 3. Similar scalability requirements exist in the metering and monitoring industries. In a network that large, it is impossible for a node to register to all Border Routers as suggested for smaller topologies in Neighbor Discovery Optimization for Low-power and Lossy Networks [I-D.ietf-6lowpan-nd].

This paper specifies IP layer functionalities that are required to implement the concept of a Backbone Router with IPv6, in particular the application of the "IP Version 6 Addressing Architecture" [RFC4291], " the Neighbor Discovery Protocol" [RFC4861] and "IPv6 Stateless Address Autoconfiguration" [RFC4862].

The use of EUI-64 based link local addresses, Neighbor Discovery Proxying [RFC4389], Neighbor Discovery Optimization for Low-power and Lossy Networks [I-D.ietf-6lowpan-nd], the IPv6 Routing Protocol for Low power and Lossy Networks [I-D.ietf-roll-rpl] and Optimistic Duplicate Address Detection [RFC4429] are discussed. Also, the concept of Transit Link is introduced to implement the backbone network that was envisioned by ISA100.11a.

This operation of the Backbone Router requires that some protocol operates over the LLNs from which node registrations can be obtained, and that can disseminate the location of the backbone Router over the LLN. Further expectations will be detailed.

The way the PAN IDs and 16-bit short addresses are allocated and distributed in the case of an 802.15.4 network is not covered by this specification. Similarly, the aspects of joining and securing the network are out of scope. The way the nodes in the LLN learn about their Backbone Router depends on the protocol used in the LLN. In the case of RPL, a Border Router is the root of the DODAG that it serves and represents all nodes attached to that DODAG.

2. Terminology

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

Readers are expected to be familiar with all the terms and concepts that are discussed in "Neighbor Discovery for IP version 6" [RFC4861], "IPv6 Stateless Address Autoconfiguration" [RFC4862], "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], "Neighbor Discovery Optimization for Low-power and Lossy Networks [I-D.ietf-6lowpan-nd] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

Readers would benefit from reading "Mobility Support in IPv6" [RFC3775], "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] and "Optimistic Duplicate Address Detection" [RFC4429] prior to this specification for a clear understanding of the art in ND-proxying and binding.

Additionally, this document uses terminology from [I-D.ietf-roll-terminology], and introduces the following terminology:

Backbone

This is an IPv6 transit link that interconnects 2 or more Backbone Routers. It is expected to be deployed as a high speed backbone in order to federate a potentially large set of LLNs. Also referred to as a LLN backbone or transit network.

Backbone Router

An IPv6 router that federates the LLN using a transit link as a backbone.

Extended LLN

This is the aggregation of multiple LLNs as defined in [RFC4919] interconnected by a Transit Link via Backbone Routers and forming a single IPv6 link.

Binding

The association of the LLN node IPv6 address and Interface ID with associated proxying states including the remaining lifetime of that association.

Registration

The process during which a LLN node injects its address in a protocol through which the Border Router can learn the address and proxy ND for it.

Primary BR

The BR that will defend a registered address for the purpose of DAD over the backbone

Secondary BR

A BR to which the address is registered. A Secondary Router MAY advertise the address over the backbone and proxy for it.

3. Overview

A Transit Link that we'll refer to as the LLN Backbone federates multiple LLNs as a single IP subnet. Each LLN in the subnet is anchored at a Backbone Router. The Backbone Routers interconnect the LLNs over that Backbone Link. A node can move freely from a LLN anchored at a Backbone Router to a LLN anchored at another Backbone Router on the same backbone and conserve its link local and any other IPv6 address it has formed.

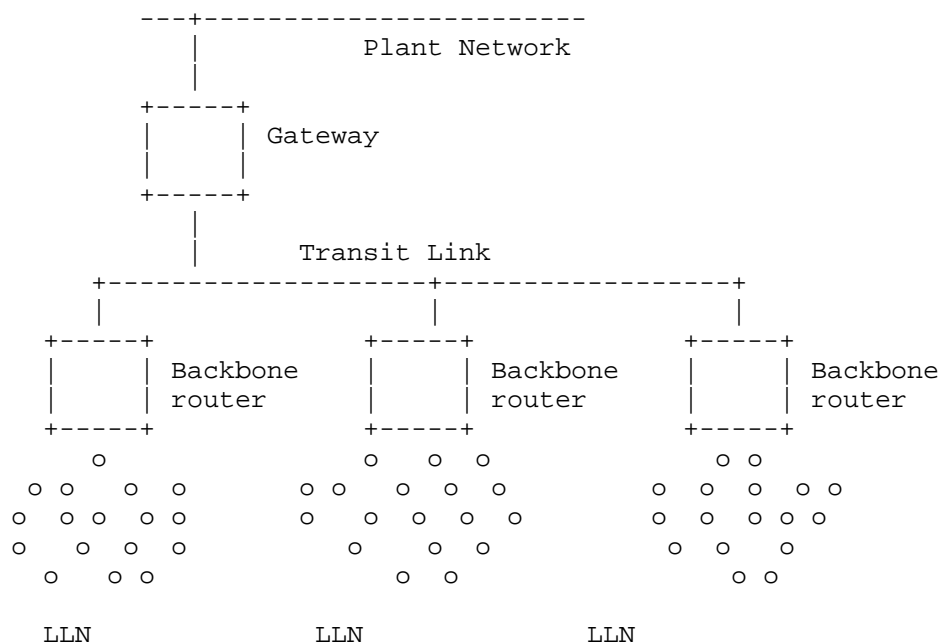


Figure 1: Backbone Link and Backbone Routers

The Backbone Link is used as reference for Neighbor Discovery operations, by extending the concept of a Home Link as defined in [RFC3775] for Mobile IPv6. In particular, Backbone Routers perform ND proxying for the LLN nodes in the LLNs they own through a node registration.

The Backbone Router operation is compatible with that of a Home Agent. This enables mobility support for LLN devices that would move outside of the network delimited by the transit link. This also enables collocation of Home Agent functionality within Backbone Router functionality on the same interface of a router.

A LLN node registers and claims ownership of its addresse(s) using proactive acknowledged registration exchanges with a neighboring router. In case of a complex LLN topology, the router might be an intermediate LLN Router that relays the registration to the LBR as described for instance in [I-D.ietf-6lowpan-nd] and [I-D.ietf-roll-rpl]. In turn, the Backbone Routers operate as a distributed database of all the LLN nodes and use the Neighbor Discovery Protocol to share that information across the transit link in a reactive fashion.

For the purpose of Neighbor Discovery proxying, this specification documents the LLN Master Neighbor Registry, a conceptual data structure that is similar to the MIP6 binding cache. The Master Neighbor Registry is fed by redistributing addresses learnt from the registration protocol used over the LLN.

Another function of the Backbone Router is to perform 6LoWPAN compression and expansion between the LLN and the Transit Link and ensure MTU compatibility. Packets flow uncompressed over the Transit Link and are routed normally towards a Gateway or an Application sitting on the transit link or on a different link that is reachable over the IP network.

4. New types and formats

The specification expects that the protocol running on the LLN can provide a sequence number called Transaction ID (TID) that is associated to the registration. When a node registers to multiple BRs, it is expected that the same TID is used, to enable the BR to correlate the registrations as being a single one, and differentiate that situation from a movement. Otherwise, the resolution makes it so that only the most recent registration was perceived from the highest TID is kept.

The specification expects that the protocol running on the LLN can provide a unique ID for the owner of the address that is being registered. The Owner Unique ID enables to differentiate a duplicate registration from a double registration. In case of a duplicate, the last registration loses. The Owner Unique ID can be as simple as a EUI-64 burnin address, if the device manufacturer is convinced that there can not be a manuf error that would cause duplicate EUI64 addresses. Alternatively, the unique ID can be a hash of supposedly unique information from multiple orthogonal sources, for instance:

- o Burn in address.
- o configured address, id, security keys...
- o (pseudo) Random number, radio link metrics ...

In any fashion, it is recommended that the device stores the unique Id in persistent memory. Otherwise, it will be prevented to reregister after a reboot that would cause a loss of memory until the Backbone Router times out the registration.

The unique ID and the sequence number are placed in a new ND option that is used by the Backbone Routers over the transit link to detect duplicates and movements. The option format is as follows:

4.1. Binding Tracking Option

This option is designed to be used with standard NS and NA messages between backbone Routers over a backbone link and may be used between LRs and LBRs over the LLN. By using this option, the binding in question can be uniquely identified and matched with the Master Neighbor Registry entries of each Backbone Router.

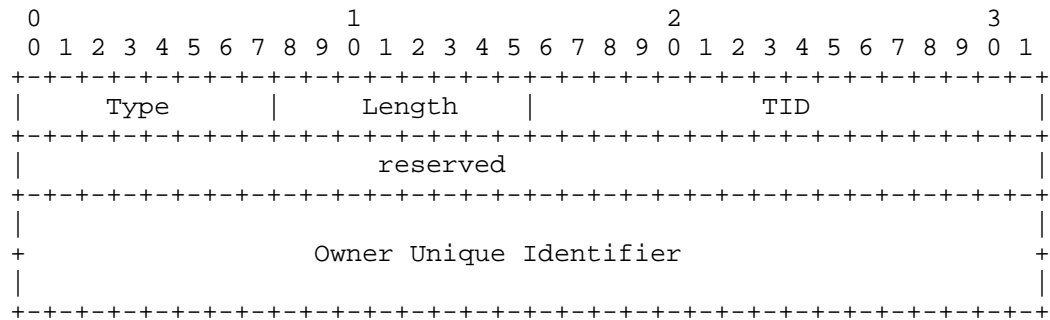


Figure 2: Binding Tracking Option

Option Fields

Type:

Length: 2

TID: A unique Transaction ID assigned by the host in the associated NR and used to match NC replies. The TID is set to zero when the node boots and then follows a lollipop lifetime, wrapping directly from 0xFFFF to 0x10.

Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Owner Unique Identifier: A globally unique identifier for the host's interface associated with the binding for the NS/NA message in question. This can be the EUI-64 derived IID of an interface, which can be hashed with other supposedly unique information from multiple orthogonal sources.

5. Backbone Router Operations

5.1. Backbone Link and Router

The Backbone Router is a specific kind of Border Router that performs proxy Neighbor Discovery on its backbone interface on behalf of the nodes that it has discovered on its Low Power Lossy Network interfaces. On the LLN side, the Backbone Router acquires its states about the nodes by terminating protocols such as RPL [I-D.ietf-roll-rpl] or 6LoWPAN ND [I-D.ietf-6lowpan-nd] as a LLN Border Router. It is expected that the backbone is the medium used to connect the subnet to the rest of the infrastructure, and that all the LBRs are connected to that backbone and support the Backbone Router feature as specified in this document.

The backbone is expected to be a high speed, reliable transit link, with affordable multicast capabilities, such as an Ethernet Network or a fully meshed NBMA network with multicast emulation, which allows a full support of classical ND as specified in [RFC4861] and subsequent RFCs. In other words, the backbone is not a LLN. Still, some restrictions of the attached LLNs will apply to the backbone. In particular, it is expected that the MTU is set to the same value on the backbone and all attached LLNs.

5.2. ND Proxy Operations

This specification enables a Backbone Router to proxy Neighbor Discovery operations over the backbone on behalf of the nodes that are registered to it, allowing any device on the backbone to reach a LLN node as if it was on-link.

In the context of this specification, proxy ND means:

- o defending a registered address over the backbone using NA messages with the Override bit set
- o advertising a registered address over the backbone using NA messages, asynchronously or as a response to a Neighbor Solicitation messages.
- o Looking up a destination over the backbone in order to deliver packets arriving from the LLN using Neighbor Solicitation messages.
- o Forwarding packets from the LLN over the backbone, and the other way around.

- o Eventually triggering a look up for a destination over the LLN that would not be registered at a given point of time, or as a verification of a registration.

The draft introduces the concept of primary and secondary BRs. The concept is defined with the granularity of an address, that is a given BR can be primary for a given address and secondary or another one, regardless on whether the addresses belong to the same node or not. The primary Backbone Router is in charge of protecting the address for DAD over the Backbone. Any of the Primary and Secondary BR may claim the address over the backbone, since they are all capable to route from the backbone to the LLN device.

When the protocol used to register the nodes over the LLN is RPL [I-D.ietf-roll-rpl], it is expected that one BR acts as virtual root coordinating LLN BRs (with the same DODAGID) over the non-LLN backbone. In that case, the virtual root may act as primary BR for all addresses that it cares to support, whereas the physical roots to which the node is attached are secondary BRs. It is also possible in a given deployment that the DODAGs are not coordinated. In that case, there is no virtual root and no secondary BR; the DODAG root is primary all the nodes registered to it over the backbone.

When the protocol used to register the nodes over the LLN is 6LoWPAN ND [I-D.ietf-6lowpan-nd], the Backbone Routers act as a distributed DAD table, using classical ND over the backbone to detect duplication. This specification requires that:

1. Registrations for all addresses that can be required to reach the device over the backbone, including registrations for IPv6 addresses based on burn-in EUI64 addresses are passed to the DAD table.
2. Nodes include the Binding Tracking Option in their NS used for registering those addresses and the LRs propagate that option to the LBRs.

A false positive duplicate detection may arise over the backbone, for instance if the node registers to more than one LBR, or if the node has moved. Both situations are handled gracefully unbeknownst to the node. In the former case, one LBR becomes primary to defend the address over the backbone while the others become secondary and may still forward packets back and forth. In the latter case the LBR that receives the newest registration wins and becomes primary.

5.3. Claiming and defending

Upon a new or an updated registration, the BR performs a DAD operation. If either a TID or a OUI is available, the BR places them in a Binding Tracking Option in all its ND messages over the backbone. If content is not available for a given field, it is set to 0.

If a primary already exists over the backbone, it will answer the DAD with an RA.

- o If a RA is received with the O bit set, the primary rejects the DAD and the DAD fails. the BR needs to inform the LLN protocol that the address is a duplicate.
- o If a RA is received with the O bit reset, the primary accepts the BR as secondary and DAD succeeds. The BR may install or maintain its proxy states for that address. This router MAY advertise the address using a NA. during a registration flow, it MAY set the O bit.
- o If no RA is received, this router assumes the role of primary and DAD succeeds. The BR may install or maintain its proxy states for that address. This router advertises the address using a NA with the O bit reset.

When the BR installs or maintains its proxy states for an address, it sends an NA with a SLLA option for that address. The Primary BR MAY set the O bit if it wished to attract the traffic for that address.

5.4. Conflict Resolution

A conflict arise when multiple BRs get a registration from a same address. This situation might arise when a node moves from a BR to another, when a node registers to multiple BRs, or in the RPL case when the BRs belong to a single coordinated DODAG.

The primary looks up the Binding Tracking Option in ND messages with a SLLA option.

- o If there is no option, normal ND operation takes place and the primary defends the address with a NA with the O bit set, adding the Binding Tracking Option with its own information.
- o If there is a Binding Tracking Option and the OUIs are different, then the conflict apparently happens between different nodes, and the the primary defends the address with a NA with the O bit set, adding the Binding Tracking Option with its own information. If

the TID in the Binding Tracking Option is in the straight part of the lollipop, it is possible that the request comes from the same node that has rebooted and formed a new OUI. The primary BR may assess its registered entry prior to answering.

- o If there is a Binding Tracking Option and the OUIs are the same:
 - * If the TID in the ND message is newer than the most recent one known by the primary router, this is interpreted as a node moving. The primary cleans up its states and stops defending the address.
 - * If the TID in the ND message is the same as the most recent one known by the primary router, this is interpreted as a double registration. In case of a DAD, the primary responds with a NA with the O bit reset, to confirm its position as primary, including the Binding Tracking Option.
 - * If the TID in the ND message is older than the most recent one known by the primary router, this is interpreted as a stale information. The primary defends the address with a NA with the O bit set, adding the Binding Tracking Option with its own information.
 - * If the TIDs are very different (more than 16 apart, discounting the straight part of the lollipop), it is impossible to resolve for sure. The primary BR should assess its registered entry prior to answering.

5.5. Assessing an entry

In a number of cases, it might happen that the information at the primary BR is stale and obsolete. In particular, a node with no permanent storage might reboot and form a different OUI, in which case the information at the BR is inaccurate and should be removed. In another case, the BR and the node have been out of reach for too long and the TID that the BR maintains is so far out that it is impossible to compare it with that stored at the BR.

In such situation, the primary Backbone Router has the possibility to assess the registration. This is performed by the protocol in use to register the node over the LLN.

When the protocol used to register the nodes over the LLN is RPL [I-D.ietf-roll-rpl], the BR sends a targetted DIS to the registered address over the registered path. A DAO back indicates that the current registration is still valid and provides the adequate data to resolve the conflict.

When the protocol used to register the nodes over the LLN is 6LoWPAN ND [I-D.ietf-6lowpan-nd], TBD.

6. Security Considerations

This specification expects that the link layer is sufficiently protected, either by means of physical or IP security for the Transit Link or MAC sublayer cryptography. In particular, it is expected that the LLN MAC provides secure unicast to/from the Backbone Router and secure broadcast from the Backbone Router in a way that prevents tempering with or replaying the RA messages.

The use of EUI-64 for forming the Interface ID in the link local address prevents the usage of Secure ND ([RFC3971] and [RFC3972]) and address privacy techniques. Considering the envisioned deployments and the MAC layer security applied, this is not considered an issue at this time.

7. IANA Considerations

A new type is requested for an ND option.

8. Acknowledgments

The author wishes to thank Zach Shelby for their help and in-depth review.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

9.2. Informative References

- [I-D.ietf-6lowpan-nd]
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low-power and Lossy Networks", draft-ietf-6lowpan-nd-09 (work in progress), April 2010.
- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., and R. Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-ietf-roll-rpl-08 (work in progress), May 2010.
- [I-D.ietf-roll-terminology]
Vasseur, J., "Terminology in Low power And Lossy

Networks", draft-ietf-roll-terminology-03 (work in progress), March 2010.

[I-D.van-beijnum-multi-mtu]

Beijnum, I., "Extensions for Multi-MTU Subnets", draft-van-beijnum-multi-mtu-02 (work in progress), February 2008.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, January 2005.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, April 2006.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

Author's Address

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

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

6LoWPAN
Internet-Draft
Intended status: Standards Track
Expires: December 6, 2010

P. Thubert, Ed.
Cisco
J. Hui
Arch Rock Corporation
June 4, 2010

LoWPAN fragment Forwarding and Recovery
draft-thubert-6lowpan-simple-fragment-recovery-07

Abstract

Considering that the IPv6 minimum MTU is 1280 bytes and that an 802.15.4 frame can have a payload limited to 74 bytes in the worst case, a packet might end up fragmented into as many as 18 fragments at the 6LoWPAN shim layer. If a single one of those fragments is lost in transmission, all fragments must be resent, further contributing to the congestion that might have caused the initial packet loss. This draft introduces a simple protocol to forward and recover individual fragments that might be lost over multiple hops between 6LoWPAN endpoints.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 6, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Rationale	4
4. Requirements	6
5. Overview	7
6. New Dispatch types and headers	7
6.1. Recoverable Fragment Dispatch type and Header	8
6.2. Fragment Acknowledgement Dispatch type and Header	8
7. Fragments Recovery	10
8. Forwarding Fragments	12
8.1. Upon the first fragment	12
8.2. Upon the next fragments	13
8.3. Upon the fragment acknowledgements	14
9. Security Considerations	14
10. IANA Considerations	14
11. Acknowledgments	14
12. References	15
12.1. Normative References	15
12.2. Informative References	15
Authors' Addresses	15

1. Introduction

In many 6LoWPAN applications, the majority of traffic is spent sending small chunks of data (order few bytes to few tens of bytes) per packet. Given that an 802.15.4 frame can carry 74 bytes or more in all cases, fragmentation is often not needed for most application traffic. However, many applications also require occasional bulk data transfer capabilities to support firmware upgrades of 6LoWPAN devices or extraction of logs from 6LoWPAN devices. In the former case, bulk data is transferred to the 6LoWPAN device, and in the latter, bulk data flows away from the 6LoWPAN device. In both cases, the bulk data size is often on the order of 10K bytes or more and end-to-end reliable transport is required.

Mechanisms such as TCP or application-layer segmentation will be used to support end-to-end reliable transport. One option to support bulk data transfer over 6LoWPAN links is to set the Maximum Segment Size to fit within the 802.15.4 MTU. Doing so, however, can add significant header overhead to each 802.15.4 frame. This causes the end-to-end transport to be aware of the delivery properties of 6LoWPAN networks, which is a layer violation.

An alternative mechanism combines the use of 6LoWPAN fragmentation in addition to transport or application-layer segmentation. Increasing the Maximum Segment Size reduces header overhead by the end-to-end transport protocol. It also encourages the transport protocol to reduce the number of outstanding datagrams, ideally to a single datagram, thus reducing the need to support out-of-order delivery common to 6LoWPAN networks.

[RFC4944] defines a datagram fragmentation mechanism for 6LoWPAN networks. However, because [RFC4944] does not define a mechanism for recovering fragments that are lost, datagram forwarding fails if even one fragment is not delivered properly to the next IP hop. End-to-end transport mechanisms will require retransmission of all fragments, wasting resources in an already resource-constrained network.

Past experience with fragmentation has shown that missassociated or lost fragments can lead to poor network behavior and, eventually, trouble at application layer. The reader is encouraged to read [RFC4963] and follow the references for more information. That experience led to the definition of the Path MTU discovery [RFC1191] protocol that limits fragmentation over the Internet.

For one-hop communications, a number of media propose a local acknowledgement mechanism that is enough to protect the fragments. In a multihop environment, an end-to-end fragment recovery mechanism

might be a good complement to a hop-by-hop MAC level recovery. This draft introduces a simple protocol to recover individual fragments between 6LoWPAN endpoints. Specifically in the case of UDP, valuable additional information can be found in UDP Usage Guidelines for Application Designers [RFC5405].

2. Terminology

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

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

ERP

Error Recovery Procedure.

LoWPAN endpoints

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

3. Rationale

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

The list of those usages includes:

Towards the LoWPAN node:

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

Firmware update: For example, a new version of the LoWPAN node software is downloaded from a system manager over unicast or multicast services. Such a reflashing operation typically involves updating a large number of similar 6LoWPAN nodes over a relatively short period of time.

From the LoWPAN node:

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

Data logs: 6LoWPAN nodes may generate large logs of sampled data for later extraction. 6LoWPAN nodes may also generate system logs to assist in diagnosing problems on the node or network.

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

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

When a fragment is lost in transmission, all fragments are resent, further contributing to the congestion that caused the initial loss, and potentially leading to congestion collapse.

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

To demonstrate the severity of the problem, consider a fairly reliable 802.15.4 frame delivery rate of 99.9% over a single 802.15.4 hop. The expected delivery rate of a 5-fragment datagram would be about 99.5% over a single 802.15.4 hop. However, the expected delivery rate would drop to 95.1% over 10 hops, a reasonable network diameter for 6LoWPAN applications. The expected delivery rate for a 1280-byte datagram is 98.4% over a single hop and 85.2% over 10 hops.

Considering that the IPv6 minimum MTU is 1280 bytes and that a 802.15.4 frame can limit the MAC payload to as little as 74 bytes, a packet might be fragmented into at least 18 fragments at the 6LoWPAN shim layer. Taking into account the worst-case header overhead for 6LoWPAN Fragmentation and Mesh Addressing headers will increase the number of required fragments to around 32. This level of fragmentation is much higher than that traditionally experienced over the Internet with IPv4 fragments. At the same time, the use of radios increases the probability of transmission loss and Mesh-Under

techniques compound that risk over multiple hops.

4. Requirements

This paper proposes a method to recover individual fragments between LoWPAN endpoints. The method is designed to fit the following requirements of a LoWPAN (with or without a Mesh-Under routing protocol):

Number of fragments

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

Minimum acknowledgement overhead

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

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

Controlled latency

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

Support for out-of-order fragment delivery

A Mesh-Under load balancing mechanism such as the ISA100 Data Link Layer can introduce out-of-sequence packets.

The recovery mechanism must account for packets that appear lost but are actually only delayed over a different path.

Optional congestion control

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

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

5. Overview

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

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

Because a meshed LoWPAN might deliver frames out of order, it is virtually impossible to differentiate these situations. In other words, from the sender standpoint, all outstanding fragments might still be in the network and contribute to its congestion. There is an assumption, though, that after a certain amount of time, a frame is either received or lost, so it is not causing congestion anymore. This amount of time can be estimated based on the round trip delay between the LoWPAN endpoints. The method detailed in [RFC2988] is recommended for that computation.

6. New Dispatch types and headers

This specification extends "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] with 4 new dispatch types, for Recoverable Fragments (RFRAG) headers with or without Acknowledgment Request, and for the Acknowledgment back.

Pattern	Header Type
11 101000	RFRAG - Recoverable Fragment
11 101001	RFRAG-AR - RFRAG with Ack Request
11 10101x	RFRAG-ACK - RFRAG Acknowledgment

Figure 1: Additional Dispatch Value Bit Patterns

In the following sections, the semantics of "datagram_tag," "datagram_offset" and "datagram_size" and the reassembly process are changed from [RFC4944] Section 5.3. "Fragmentation Type and Header." The size and offset are expressed on the compressed packet as opposed to the uncompressed form.

6.1. Recoverable Fragment Dispatch type and Header

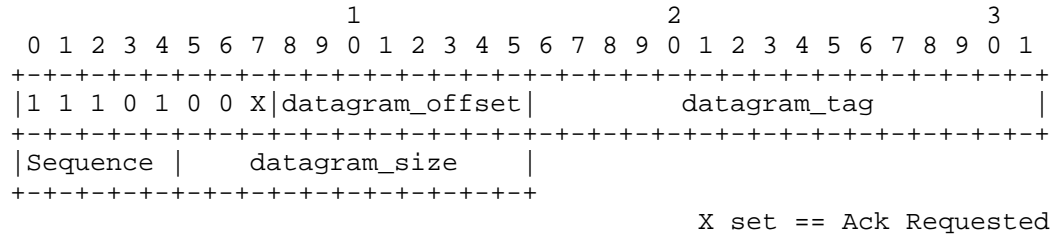


Figure 2: Recoverable Fragment Dispatch type and Header

X bit

When set, the sender requires an Acknowledgment from the receiver

Sequence

The sequence number of the fragment. Fragments are numbered [0..N] where N is in [0..31].

6.2. Fragment Acknowledgement Dispatch type and Header

The specification also defines an acknowledgement bitmap that is used to carry selective acknowledgements for the received fragments. A given offset in the bitmap maps one to one with a given sequence number.

The bitmap is compressed as a variable length field formed by control bits and acknowledgement bits. The leftmost bits of the compressed form are control bits up to the first 0. The rest is ack bits encoded right to left:

Pattern	Size	Ack
0XXXXXXXX	1 octet	1 -> 7
10XXXXXX XXXXXXXX	2 octets	1 -> 14
110XXXXX XXXXXXXX XXXXXXXX	3 octets	1 -> 21
1110XXXX XXXXXXXX XXXXXXXX XXXXXXXX	4 octets	1 -> 28

Figure 3: Compressed acknowledgement bitmap encoding

The highest sequence number to be acknowledged determines the pattern to be used. The format can be extended for more fragments in the future but this specification only requires the support of up to 4 octets encoding, which enables to acknowledge up to 28 fragments.

A 32 bits uncompressed bitmap is obtained by prepending zeroes to the XXX in the pattern above. For instance:

```

  0 1 2 3 4 5 6 7
+-----+
|0|1|1|0|1|1|1|1|  is expanded as:
+-----+

          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|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|1|0|1|1|1|1|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 4: Expanding 1 octet encoding

and

```

          1                      2
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-----+-----+-----+-----+-----+-----+
|1|1|0|1|1|1|1|0|1|1|1|1|1|1|1|1|1|1|1|1|0|0|1|  is expanded as:
+-----+-----+-----+-----+-----+-----+

          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|0|0|0|0|0|0|0|0|0|0|0|1|1|1|1|0|1|1|1|1|1|1|1|1|1|1|1|1|0|0|1|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 5: Expanding 3 octets encoding

whereas the 4 octets encoding is expanded by simply setting the first 3 bits to 0. The 32 bits uncompressed bitmap is written and read 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
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Acknowledgment Bitmap               |
+-----+-----+-----+-----+-----+-----+-----+-----+
                                     ^                     ^
      bitmap indicating whether:      |                     |
      Fragment with sequence 10 was received --+         |
      Fragment with sequence 00 was received -----+

```

Figure 6: Expanded bitmap encoding

So in the example in Figure 5 it appears that all fragments from sequence 0 to 20 were received but for sequence 1, 2 and 16 that were either lost or are still in the network over a slower path.

The compressed form of the acknowledgement bitmap is carried in a Fragment Acknowledgement 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
      +-----+-----+-----+-----+-----+-----+-----+
      |1 1 1 0 1 0 1 Y|          datagram_tag          |
+-----+-----+-----+-----+-----+-----+-----+
| Compressed Acknowledgment Bitmap (8 to 32 bits)
+-----+-----+-----+

```

Figure 7: Fragment Acknowledgement Dispatch type and Header

Y bit

Reserved for Explicit Congestion Notification (ECN) signalling

Compressed Acknowledgement Bitmap

An encoded form of an acknowledgement bitmap.

7. Fragments Recovery

The Recoverable Fragments header RFRAG and RFRAG-AR deprecate the original fragment headers from [RFC4944] and replace them in the fragmented packets. The Fragment Acknowledgement RFRAG-ACK is introduced as a standalone header in message that is sent back to the fragment source endpoint as known by its MAC address. This assumes that the source MAC address in the fragment (is any) and datagram_tag are enough information to send the Fragment Acknowledgement back to the source fragmentation endpoint.

The node that fragments the packets at 6LoWPAN level (the sender) controls the Fragment Acknowledgements. It may do that at any fragment to implement its own policy or perform congestion control which is out of scope for this document. When the sender of the fragment knows that an underlying mechanism protects the Fragments already it MAY refrain from using the Acknowledgement mechanism, and never set the Ack Requested bit. The node that recomposes the packets at 6LoWPAN level (the receiver) MUST acknowledge the fragments it has received when asked to, and MAY slightly defer that acknowledgement.

The sender transfers a controlled number of fragments and MAY flag the last fragment of a series with an acknowledgment request. The receiver MUST acknowledge a fragment with the acknowledgment request bit set. If any fragment immediately preceding an acknowledgment

request is still missing, the receiver MAY intentionally delay its acknowledgment to allow in-transit fragments to arrive. delaying the acknowledgement might defeat the round trip delay computation so it should be configurable and not enabled by default.

The receiver interacts with the sender using an Acknowledgment message with a bitmap that indicates which fragments were actually received. The bitmap is a 32bit SWORD, which accommodates up to 32 fragments and is sufficient for the 6LoWPAN MTU. For all n in $[0..31]$, bit n is set to 1 in the bitmap to indicate that fragment with sequence n was received, otherwise the bit is set to 0. All zeroes is a NULL bitmap that indicates that the fragmentation process was cancelled by the receiver for that datagram.

The receiver MAY issue unsolicited acknowledgments. An unsolicited acknowledgment enables the sender endpoint to resume sending if it had reached its maximum number of outstanding fragments or indicate that the receiver has cancelled the process of an individual datagram. Note that acknowledgments might consume precious resources so the use of unsolicited acknowledgments should be configurable and not enabled by default.

The sender arms a retry timer to cover the fragment that carries the Acknowledgment request. Upon time out, the sender assumes that all the fragments on the way are received or lost. The process must have completed within an acceptable time that is within the boundaries of upper layer retries. The method detailed in [RFC2988] is recommended for the computation of the retry timer. It is expected that the upper layer retries obey the same or friendly rules in which case a single round of fragment recovery should fit within the upper layer recovery timers.

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

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

The receiver might need to cancel the process of a fragmented packet for internal reasons, for instance if it is out of recomposition

buffers, or considers that this packet is already fully recomposed and passed to the upper layer. In that case, the receiver SHOULD indicate so to the sender with a NULL bitmap. Upon an acknowledgement with a NULL bitmap, the sender MUST drop the datagram.

8. Forwarding Fragments

This specification enables intermediate routers to forward fragments with no intermediate reconstruction of the entire packet. Upon the first fragment, the routers lay an label along the path that is followed by that fragment (that is IP routed), and all further fragments are label switched along that path. As a consequence, alternate routes not possible for individual fragments. The datagram tag is used to carry the label, that is swapped at each hop.

8.1. Upon the first fragment

In route over the L2 source changes at each hop. The label that is formed and placed in the datagram tag is associated to the source MAC and only valid (and unique) for that source MAC. Say the first fragment has:

Source IPv6 address = IP_A (maybe hops away)

Destination IPv6 address = IP_B (maybe hops away)

Source MAC = MAC_prv (prv as previous)

Datagram_tag= DT_prv

The intermediate router that forwards individual fragments does the following:

a route lookup to get Next hop IPv6 towards IP_B, which resolves as IP_nxt (nxt as next)

a ND resolution to get the MAC address associated to IP_nxt, which resolves as MAC_nxt

Since it is a first fragment of a packet from that source MAC address MAC_prv for that tag DT_prv, the router:

cleans up any leftover resource associated to the tuple (MAC_prv, DT_prv)

allocates a new label for that flow, DT_nxt, from a Least Recently Used pool or some similar procedure.

allocates a Label swap structure indexed by (MAC_prv, DT_prv) that contains (MAC_nxt, DT_nxt)

allocates a Label swap structure indexed by (MAC_nxt, DT_nxt) that contains (MAC_prv, DT_prv)

swaps the MAC info to from self to MAC_nxt

Swaps the datagram_tag to DT_nxt

At this point the router is all set and can forward the packet to nxt.

8.2. Upon the next fragments

Upon next fragments (that are not first fragment), the router expects to have already Label swap structure indexed by (MAC_prv, DT_prv). The router:

lookups up the Label swap entry for (MAC_prv, DT_prv), which resolves as (MAC_nxt, DT_nxt)

swaps the MAC info to from self to MAC_nxt;

Swaps the datagram_tag to DT_nxt

At this point the router is all set and can forward the packet to nxt.

if the Label swap entry for (MAC_src, DT_src) is not found, the router builds an RFRAG-ACK to indicate the error. The acknowledgment message has the following information:

MAC info set to from self to MAC_prv as found in the fragment

Swaps the datagram_tag set to DT_prv

Bitmap of all zeroes to indicate the error

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

8.3. Upon the fragment acknowledgements

Upon fragment acknowledgements next fragments (that are not first fragment), the router expects to have already Label swap structure indexed by (MAC_nxt, DT_nxt). The router:

lookups up the Label swap entry for (MAC_nxt, DT_nxt), which resolves as (MAC_prv, DT_prv)

swaps the MAC info to from self to MAC_prv;

Swaps the datagram_tag to DT_prv

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

if the Label swap entry for (MAC_nxt, DT_nxt) is not found, it simply drops the packet.

if the RFRAG-ACK indicates either an error or that the fragment was fully receive, the router schedules the Label swap entries for recycling. If the RFRAG-ACK is lost on the way back, the source may retry the last fragments, which will result as an error RFRAG-ACK from the first router on the way that has already cleaned up.

9. Security Considerations

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

10. IANA Considerations

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

11. Acknowledgments

The author wishes to thank Jay Werb, Christos Polyzois, Soumitri Kolavennu and Harry Courtice for their contribution and review.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

12.2. Informative References

- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., and R. Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks",
draft-ietf-roll-rpl-08 (work in progress), May 2010.
- [I-D.mathis-frag-harmful]
Mathis, M., "Fragmentation Considered Very Harmful",
draft-mathis-frag-harmful-00 (work in progress),
July 2004.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
November 1990.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals",
RFC 4919, August 2007.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405,
November 2008.

Authors' Addresses

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

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

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

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

