

6Lo Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 5, 2016

J. Nieminen
T. Savolainen
M. Isomaki
Nokia
B. Patil
AT&T
Z. Shelby
Arm
C. Gomez
Universitat Politecnica de Catalunya/i2CAT
August 4, 2015

IPv6 over BLUETOOTH(R) Low Energy
draft-ietf-6lo-btle-17

Abstract

Bluetooth Smart is the brand name for the Bluetooth low energy feature in the Bluetooth specification defined by the Bluetooth Special Interest Group. The standard Bluetooth radio has been widely implemented and available in mobile phones, notebook computers, audio headsets and many other devices. The low power version of Bluetooth is a specification that enables the use of this air interface with devices such as sensors, smart meters, appliances, etc. The low power variant of Bluetooth has been standardized since revision 4.0 of the Bluetooth specifications, although version 4.1 or newer is required for IPv6. This document describes how IPv6 is transported over Bluetooth low energy using IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) techniques.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 5, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Terminology and Requirements Language	3
2. Bluetooth Low Energy	3
2.1. Bluetooth LE stack	4
2.2. Link layer roles and topology	5
2.3. Bluetooth LE device addressing	6
2.4. Bluetooth LE packet sizes and MTU	6
3. Specification of IPv6 over Bluetooth Low Energy	7
3.1. Protocol stack	7
3.2. Link model	8
3.2.1. IPv6 subnet model and Internet connectivity	9
3.2.2. Stateless address autoconfiguration	10
3.2.3. Neighbor discovery	12
3.2.4. Header compression	13
3.2.4.1. Remote destination example	14
3.2.4.2. Example of registration of multiple-addresses	15
3.2.5. Unicast and Multicast address mapping	16
4. IANA Considerations	16
5. Security Considerations	16
6. Additional contributors	17
7. Acknowledgements	17
8. References	18
8.1. Normative References	18
8.2. Informative References	19
Authors' Addresses	20

1. Introduction

Bluetooth Smart is the brand name for the Bluetooth low energy feature (hereinafter, Bluetooth LE) in the Bluetooth specification defined by the Bluetooth Special Interest Group. Bluetooth LE is a

radio technology targeted for devices that operate with very low capacity (e.g., coin cell) batteries or minimalistic power sources, which means that low power consumption is essential. Bluetooth LE is especially attractive technology for Internet of Things applications, such as health monitors, environmental sensing, proximity applications and many others.

Considering the potential for the exponential growth in the number of sensors and Internet connected devices, IPv6 is an ideal protocol for communication with such devices due to the large address space it provides. In addition, IPv6 provides tools for stateless address autoconfiguration, which is particularly suitable for sensor network applications and nodes which have very limited processing power or lack a full-fledged operating system.

This document describes how IPv6 is transported over Bluetooth LE connections using IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) techniques. RFCs 4944, 6282, and 6775 [RFC4944][RFC6282][RFC6775] developed for 6LoWPAN specify the transmission of IPv6 over IEEE 802.15.4 [fifteendotfour]. The Bluetooth LE link in many respects has similar characteristics to that of IEEE 802.15.4 and many of the mechanisms defined for the IPv6 over IEEE 802.15.4 can be applied to the transmission of IPv6 on Bluetooth LE links. This document specifies the details of IPv6 transmission over Bluetooth LE links.

1.1. Terminology and Requirements Language

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

The terms 6LoWPAN Node (6LN), 6LoWPAN Router (6LR) and 6LoWPAN Border Router (6LBR) are defined as in [RFC6775], with an addition that Bluetooth LE central and Bluetooth LE peripheral (see Section 2.2) can both be either 6LN or 6LBR.

2. Bluetooth Low Energy

Bluetooth LE is designed for transferring small amounts of data infrequently at modest data rates with a very small energy expenditure per bit. Bluetooth Special Interest Group (Bluetooth SIG) has introduced two trademarks, Bluetooth Smart for single-mode devices (a device that only supports Bluetooth LE) and Bluetooth Smart Ready for dual-mode devices (devices that support both Bluetooth and Bluetooth LE; note that Bluetooth and Bluetooth LE are different, non-interoperable radio technologies). In the rest of the

document, the term Bluetooth LE is used regardless of whether this technology is supported by a single-mode or dual-mode device.

Bluetooth LE was introduced in Bluetooth 4.0, enhanced in Bluetooth 4.1 [BTCorev4.1], and developed even further in successive versions. Bluetooth SIG has also published the Internet Protocol Support Profile (IPSP) [IPSP], which includes the Internet Protocol Support Service (IPSS). The IPSP enables discovery of IP-enabled devices and establishment of a link layer connection for transporting IPv6 packets. IPv6 over Bluetooth LE is dependent on both Bluetooth 4.1 and IPSP 1.0 or more recent versions of either specification to provide necessary capabilities.

Devices such as mobile phones, notebooks, tablets and other handheld computing devices that incorporate chipsets implementing Bluetooth 4.1 or later will also have the low-energy functionality of Bluetooth. Bluetooth LE is also expected to be included in many different types of accessories that collaborate with mobile devices such as phones, tablets and notebook computers. An example of a use case for a Bluetooth LE accessory is a heart rate monitor that sends data via the mobile phone to a server on the Internet.

2.1. Bluetooth LE stack

The lower layer of the Bluetooth LE stack consists of the Physical (PHY), the Link Layer (LL), and a test interface called the Direct Test Mode (DTM). The Physical Layer transmits and receives the actual packets. The Link Layer is responsible for providing medium access, connection establishment, error control and flow control. The Direct Test Mode is only used for testing purposes. The upper layer consists of the Logical Link Control and Adaptation Protocol (L2CAP), Attribute Protocol (ATT), Security Manager (SM), Generic Attribute Profile (GATT) and Generic Access Profile (GAP) as shown in Figure 1. The Host Controller Interface (HCI) separates the lower layers, often implemented in the Bluetooth controller, from higher layers, often implemented in the host stack. GATT and Bluetooth LE profiles together enable the creation of applications in a standardized way without using IP. L2CAP provides multiplexing capability by multiplexing the data channels from the above layers. L2CAP also provides fragmentation and reassembly for large data packets. The Security Manager defines a protocol and mechanisms for pairing, key distribution and a security toolbox for the Bluetooth LE device.

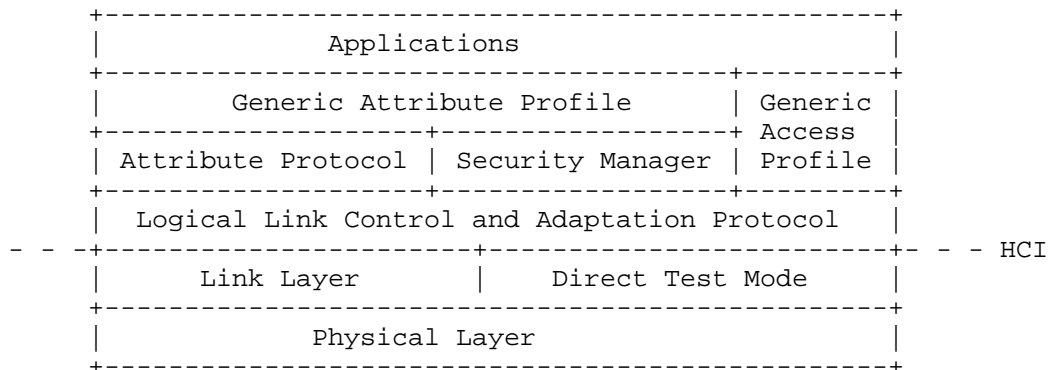


Figure 1: Bluetooth LE Protocol Stack

As shown in Section 3.1, IPv6 over Bluetooth LE requires an adapted 6LoWPAN layer which runs on top of Bluetooth LE L2CAP.

2.2. Link layer roles and topology

Bluetooth LE defines two GAP roles of relevance herein: the Bluetooth LE central role and the Bluetooth LE peripheral role. A device in the central role, which is called central from now on, has traditionally been able to manage multiple simultaneous connections with a number of devices in the peripheral role, called peripherals from now on. A peripheral is commonly connected to a single central, but with versions of Bluetooth from 4.1 onwards it can also connect to multiple centrals at the same time. In this document for IPv6 networking purposes the Bluetooth LE network (i.e., a Bluetooth LE piconet) follows a star topology shown in the Figure 2, where a router typically implements the Bluetooth LE central role and the rest of nodes implement the Bluetooth LE peripheral role. In the future mesh networking and/or parallel connectivity to multiple centrals at a time may be defined for IPv6 over Bluetooth LE.

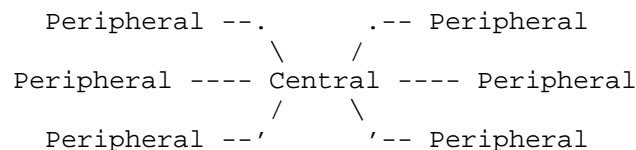


Figure 2: Bluetooth LE Star Topology

In Bluetooth LE, direct wireless communication only takes place between a central and a peripheral. This means that inherently the Bluetooth LE star represents a hub and spokes link model.

Nevertheless, two peripherals may communicate through the central by using IP routing functionality per this specification.

2.3. Bluetooth LE device addressing

Every Bluetooth LE device is identified by a 48-bit device address. The Bluetooth specification describes the device address of a Bluetooth LE device as: "Devices are identified using a device address. Device addresses may be either a public device address or a random device address." [BTCorev4.1]. The public device addresses are based on the IEEE 802-2001 standard [IEEE802-2001]. Random device addresses and Bluetooth LE privacy feature are described in Bluetooth Generic Access Profile specification sections 10.8 and 10.7, respectively [BTCorev4.1]. There are two types of random device addresses: static and private addresses. The private addresses are further divided into two sub-types: resolvable or non-resolvable addresses, which are explained in depth in the referenced Bluetooth specification. Once a static address is initialized, it does not change until the device is power cycled. The static address can be initialized to a new value after each power cycle, but that is not mandatory. Recommended time interval before randomizing new private address is 15 minutes, as determined by timer T_GAP(private_addr_int) at Bluetooth Generic Access Profile Table 17.1. The selection of which device address types are used is implementation and deployment specific. In random addresses first 46 bits are randomized and last 2 bits indicate the random address type. Bluetooth LE does not support device address collision avoidance or detection. However, these 48 bit random device addresses have a very small probability of being in conflict within a typical deployment.

2.4. Bluetooth LE packet sizes and MTU

The optimal MTU defined for L2CAP fixed channels over Bluetooth LE is 27 octets including the L2CAP header of 4 octets. The default MTU for Bluetooth LE is hence defined to be 27 octets. Therefore, excluding the L2CAP header of 4 octets, a protocol data unit (PDU) size of 23 octets is available for upper layers. In order to be able to transmit IPv6 packets of 1280 octets or larger, a link layer fragmentation and reassembly solution is provided by the L2CAP layer. The IPSP defines means for negotiating up a link layer connection that provides an MTU of 1280 octets or higher for the IPv6 layer [IPSP]. The link layer MTU is negotiated separately for each direction. Implementations that require an equal link layer MTU for the two directions SHALL use the smallest of the possibly different MTU values.

3. Specification of IPv6 over Bluetooth Low Energy

Bluetooth LE technology sets strict requirements for low power consumption and thus limits the allowed protocol overhead. 6LoWPAN standards [RFC6775], and [RFC6282] provide useful functionality for reducing overhead, which are applied to Bluetooth LE. This functionality is comprised of link-local IPv6 addresses and stateless IPv6 address autoconfiguration (see Section 3.2.2), Neighbor Discovery (see Section 3.2.3), and header compression (see Section 3.2.4). Fragmentation features from 6LoWPAN standards are not used due to Bluetooth LE's link layer fragmentation support (see Section 2.4).

A significant difference between IEEE 802.15.4 and Bluetooth LE is that the former supports both star and mesh topologies (and requires a routing protocol), whereas Bluetooth LE does not currently support the formation of multihop networks at the link layer. However, inter-peripheral communication through the central is enabled by using IP routing functionality per this specification.

In Bluetooth LE a central node is assumed to be less resource constrained than a peripheral node. Hence, in the primary deployment scenario central and peripheral will act as 6LoWPAN Border Router (6LBR) and a 6LoWPAN Node (6LN), respectively.

Before any IP-layer communications can take place over Bluetooth LE, Bluetooth LE enabled nodes such as 6LNs and 6LBRs have to find each other and establish a suitable link layer connection. The discovery and Bluetooth LE connection setup procedures are documented by the Bluetooth SIG in the IPSP specification [IPSP].

In the rare case of Bluetooth LE random device address conflict, a 6LBR can detect multiple 6LNs with the same Bluetooth LE device address, as well as a 6LN with the same Bluetooth LE address as the 6LBR. The 6LBR MUST ignore 6LNs with the same device address the 6LBR has, and the 6LBR MUST have at most one connection for a given Bluetooth LE device address at any given moment. This will avoid addressing conflicts within a Bluetooth LE network.

3.1. Protocol stack

Figure 3 illustrates how the IPv6 stack works in parallel to the GATT stack on top of Bluetooth LE L2CAP layer. The GATT stack is needed herein for discovering nodes supporting the Internet Protocol Support Service. UDP and TCP are provided as examples of transport protocols, but the stack can be used by any other upper layer protocol capable of running atop of IPv6.

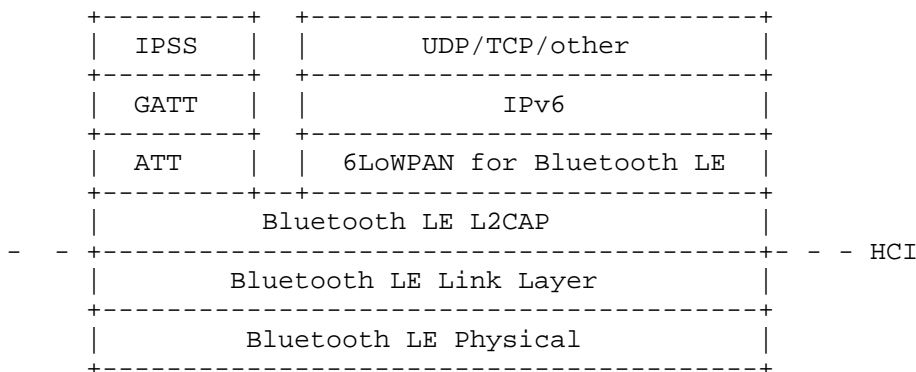


Figure 3: IPv6 and IPSS on the Bluetooth LE Stack

3.2. Link model

The distinct concepts of the IPv6 link (layer 3) and the physical link (combination of PHY and MAC) need to be clear and their relationship has to be well understood in order to specify the addressing scheme for transmitting IPv6 packets over the Bluetooth LE link. RFC 4861 [RFC4861] defines a link as "a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6."

In the case of Bluetooth LE, the 6LoWPAN layer is adapted to support transmission of IPv6 packets over Bluetooth LE. The IPSP defines all steps required for setting up the Bluetooth LE connection over which 6LoWPAN can function [IPSP], including handling the link layer fragmentation required on Bluetooth LE, as described in Section 2.4. Even though MTUs larger than 1280 octets can be supported, use of a 1280 octet MTU is RECOMMENDED in order to avoid need for Path MTU discovery procedures.

While Bluetooth LE protocols, such as L2CAP, utilize little-endian byte ordering, IPv6 packets MUST be transmitted in big endian order (network byte order).

Per this specification, the IPv6 header compression format specified in RFC 6282 MUST be used [RFC6282]. The IPv6 payload length can be derived from the L2CAP header length and the possibly elided IPv6 address can be reconstructed from the link layer address, used at the time of Bluetooth LE connection establishment, from the HCI Connection Handle during connection, compression context if any, and from address registration information (see Section 3.2.3).

Bluetooth LE connections used to build a star topology are point-to-point in nature, as Bluetooth broadcast features are not used for IPv6 over Bluetooth LE (except for discovery of nodes supporting IPSS). After the peripheral and central have connected at the Bluetooth LE level, the link can be considered up and IPv6 address configuration and transmission can begin.

3.2.1. IPv6 subnet model and Internet connectivity

In the Bluetooth LE piconet model (see Section 2.2) peripherals each have a separate link to the central and the central acts as an IPv6 router rather than a link layer switch. As discussed in [RFC4903], conventional usage of IPv6 anticipates IPv6 subnets spanning a single link at the link layer. As IPv6 over Bluetooth LE is intended for constrained nodes, and for Internet of Things use cases and environments, the complexity of implementing a separate subnet on each peripheral-central link and routing between the subnets appears to be excessive. In the Bluetooth LE case, the benefits of treating the collection of point-to-point links between a central and its connected peripherals as a single multilink subnet rather than a multiplicity of separate subnets are considered to outweigh the multilink model's drawbacks as described in [RFC4903].

Hence a multilink model has been chosen, as further illustrated in Figure 4. Because of this, link-local multicast communications can happen only within a single Bluetooth LE connection, and thus 6LN-to-6LN communications using link-local addresses are not possible. 6LNs connected to the same 6LBR have to communicate with each other by using the shared prefix used on the subnet. The 6LBR ensures address collisions do not occur (see Section 3.2.3) and forwards packets sent by one 6LN to another.

In a typical scenario, the Bluetooth LE network is connected to the Internet as shown in the Figure 4. In this scenario, the Bluetooth LE star is deployed as one subnet, using one /64 IPv6 prefix, with each spoke representing individual link. The 6LBR is acting as router and forwarding packets between 6LNs and to and from Internet.

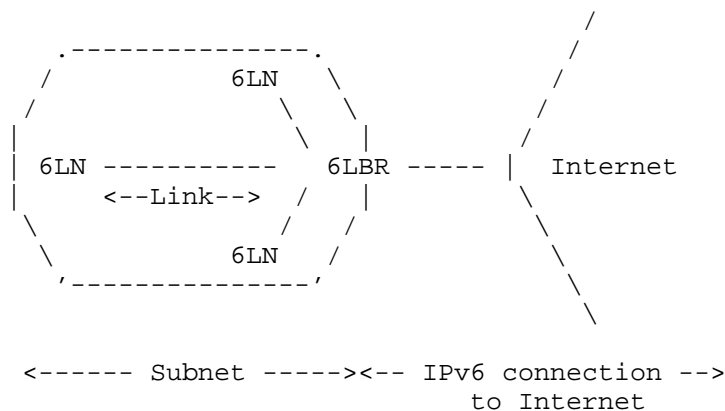


Figure 4: Bluetooth LE network connected to the Internet

In some scenarios, the Bluetooth LE network may transiently or permanently be an isolated network as shown in the Figure 5. In this case the whole star consist of a single subnet with multiple links, where 6LBR is at central routing packets between 6LNs. In simplest case the isolated network has one 6LBR and one 6LN.

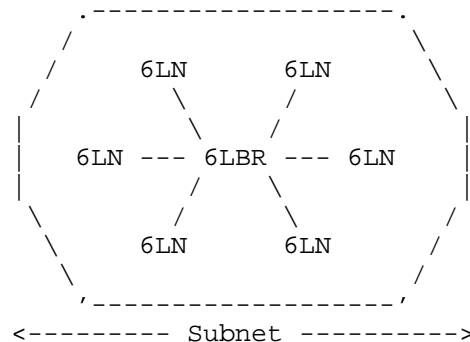


Figure 5: Isolated Bluetooth LE network

3.2.2. Stateless address autoconfiguration

At network interface initialization, both 6LN and 6LBR SHALL generate and assign to the Bluetooth LE network interface IPv6 link-local addresses [RFC4862] based on the 48-bit Bluetooth device addresses (see Section 2.3) that were used for establishing the underlying Bluetooth LE connection. A 6LN and a 6LBR are RECOMMENDED to use private Bluetooth device addresses. A 6LN SHOULD pick a different

Bluetooth device address for every Bluetooth LE connection with a 6LBR, and a 6LBR SHOULD periodically change its random Bluetooth device address. Following the guidance of [RFC7136], a 64-bit Interface Identifier (IID) is formed from the 48-bit Bluetooth device address by inserting two octets, with hexadecimal values of 0xFF and 0xFE in the middle of the 48-bit Bluetooth device address as shown in Figure 6. In the Figure letter 'b' represents a bit from the Bluetooth device address, copied as is without any changes on any bit. This means that no bit in the IID indicates whether the underlying Bluetooth device address is public or random.

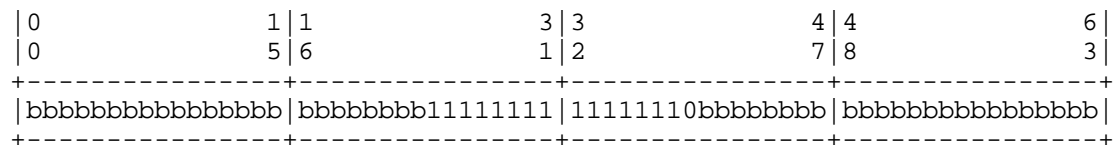


Figure 6: Formation of IID from Bluetooth device address

The IID is then prepended with the prefix fe80::/64, as described in RFC 4291 [RFC4291] and as depicted in Figure 7. The same link-local address SHALL be used for the lifetime of the Bluetooth LE L2CAP channel. (After a Bluetooth LE logical link has been established, it is referenced with a Connection Handle in HCI. Thus possibly changing device addresses do not impact data flows within existing L2CAP channels. Hence there is no need to change IPv6 link-local addresses even if devices change their random device addresses during L2CAP channel lifetime).

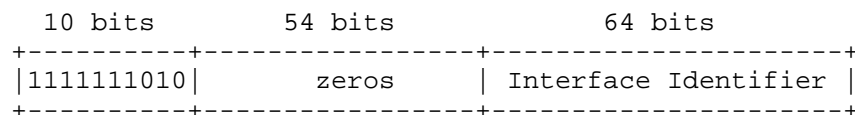


Figure 7: IPv6 link-local address in Bluetooth LE

A 6LN MUST join the all-nodes multicast address. There is no need for 6LN to join the solicited-node multicast address, since 6LBR will know device addresses and hence link-local addresses of all connected 6LNs. The 6LBR will ensure no two devices with the same Bluetooth LE device address are connected at the same time. Detection of duplicate link-local addresses is performed by the process on the 6LBR responsible for the discovery of IP-enabled Bluetooth LE nodes and for starting Bluetooth LE connection establishment procedures.

This approach increases the complexity of 6LBR, but reduces power consumption on both 6LN and 6LBR in the link establishment phase by reducing the number of mandatory packet transmissions.

After link-local address configuration, the 6LN sends Router Solicitation messages as described in [RFC4861] Section 6.3.7.

For non-link-local addresses, 6LNs SHOULD NOT be configured to embed the Bluetooth device address in the IID by default. Alternative schemes such as Cryptographically Generated Addresses (CGA) [RFC3972], privacy extensions [RFC4941], Hash-Based Addresses (HBA, [RFC5535]), DHCPv6 [RFC3315], or static, semantically opaque addresses [RFC7217] SHOULD be used by default. In situations where the Bluetooth device address is known to be a private device address and/or the header compression benefits of embedding the device address in the IID are required to support deployment constraints, 6LNs MAY form a 64-bit IID by utilizing the 48-bit Bluetooth device address. The non-link-local addresses that a 6LN generates MUST be registered with the 6LBR as described in Section 3.2.3.

The tool for a 6LBR to obtain an IPv6 prefix for numbering the Bluetooth LE network is out of scope of this document, but can be, for example, accomplished via DHCPv6 Prefix Delegation [RFC3633] or by using Unique Local IPv6 Unicast Addresses (ULA) [RFC4193]. Due to the link model of the Bluetooth LE (see Section 3.2.1) the 6LBR MUST set the "on-link" flag (L) to zero in the Prefix Information Option in Neighbor Discovery messages [RFC4861] (see Section 3.2.3). This will cause 6LNs to always send packets to the 6LBR, including the case when the destination is another 6LN using the same prefix.

3.2.3. Neighbor discovery

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

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

1. A Bluetooth LE 6LN MUST NOT register its link-local address. A Bluetooth LE 6LN MUST register its non-link-local addresses with the 6LBR by sending a Neighbor Solicitation (NS) message with the Address Registration Option (ARO) and process the Neighbor Advertisement (NA) accordingly. The NS with the ARO option MUST be sent irrespective of the method used to generate the IID. If the 6LN registers for a same

compression context multiple addresses that are not based on Bluetooth device address, the header compression efficiency will decrease (see Section 3.2.4).

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

3.2.4. Header compression

Header compression as defined in RFC 6282 [RFC6282], which specifies the compression format for IPv6 datagrams on top of IEEE 802.15.4, is REQUIRED as the basis for IPv6 header compression on top of Bluetooth LE. All headers MUST be compressed according to RFC 6282 [RFC6282] encoding formats.

The Bluetooth LE's star topology structure and ARO can be exploited in order to provide a mechanism for address compression. The following text describes the principles of IPv6 address compression on top of Bluetooth LE.

The ARO option requires use of an EUI-64 identifier [RFC6775]. In the case of Bluetooth LE, the field SHALL be filled with the 48-bit device address used by the Bluetooth LE node converted into 64-bit Modified EUI-64 format [RFC4291].

To enable efficient header compression, when the 6LBR sends a Router Advertisement it MUST include a 6LoWPAN Context Option (6CO) [RFC6775] matching each address prefix advertised via a Prefix Information Option (PIO) [RFC4861] for use in stateless address autoconfiguration.

When a 6LN is sending a packet to a 6LBR, it MUST fully elide the source address if it is a link-local address. For other packets to or through a 6LBR with a non-link-local source address that the 6LN has registered with ARO to the 6LBR for the indicated prefix, the source address MUST be fully elided if it is the latest address that the 6LN has registered for the indicated prefix. If a source non-link-local address is not the latest registered, then the 64-bits of the IID SHALL be fully carried in-line (SAM=01) or if the first 48-bits of the IID match with the latest registered address, then the last 16-bits of the IID SHALL be carried in-line (SAM=10). That is, if SAC=0 and SAM=11 the 6LN MUST be using the link-local IPv6 address derived from Bluetooth LE device address, and if SAC=1 and SAM=11 the 6LN MUST have registered the source IPv6 address with the prefix related to the compression context and the 6LN MUST be referring to the latest registered address related to the compression context. The IPv6 address MUST be considered to be registered only after the

6LBR has sent a Neighbor Advertisement with an ARO having its status field set to success. The destination IPv6 address MUST be fully elided if the destination address is 6LBR's link-local-address based on the 6LBR's Bluetooth device address (DAC=0, DAM=11). The destination IPv6 address MUST be fully or partially elided if context has been set up for the destination address. For example, DAC=0 and DAM=01 when destination prefix is link-local, and DAC=1 and DAM=01 if compression context has been configured for the destination prefix used.

When a 6LBR is transmitting packets to a 6LN, it MUST fully elide the source IID if the source IPv6 address is the link-local address based on the 6LBR's Bluetooth device address (SAC=0, SAM=11), and it MUST elide the source prefix or address if a compression context related to the IPv6 source address has been set up. The 6LBR also MUST fully elide the destination IPv6 address if it is the link-local-address based on the 6LN's Bluetooth device address (DAC=0, DAM=11), or if the destination address is the latest registered by the 6LN with ARO for the indicated context (DAC=1, DAM=11). If the destination address is a non-link-local address and not the latest registered, then the 6LN MUST either include the IID part fully in-line (DAM=01) or, if the first 48-bits of the IID match to the latest registered address, then elide those 48-bits (DAM=10).

3.2.4.1. Remote destination example

When a 6LN transmits an IPv6 packet to a remote destination using global Unicast IPv6 addresses, if a context is defined for the 6LN's global IPv6 address, the 6LN has to indicate this context in the corresponding source fields of the compressed IPv6 header as per Section 3.1 of RFC 6282 [RFC6282], and has to elide the full IPv6 source address previously registered with ARO (if using the latest registered address, otherwise part or all of the IID may have to be transmitted in-line). For this, the 6LN MUST use the following settings in the IPv6 compressed header: SAC=1 and SAM=11. The CID may be set 0 or 1, depending on which context is used. In this case, the 6LBR can infer the elided IPv6 source address since 1) the 6LBR has previously assigned the prefix to the 6LNs; and 2) the 6LBR maintains a Neighbor Cache that relates the Device Address and the IID the device has registered with ARO. If a context is defined for the IPv6 destination address, the 6LN has to also indicate this context in the corresponding destination fields of the compressed IPv6 header, and elide the prefix of or the full destination IPv6 address. For this, the 6LN MUST set the DAM field of the compressed IPv6 header as DAM=01 (if the context covers a 64-bit prefix) or as DAM=11 (if the context covers a full, 128-bit address). DAC MUST be set to 1. Note that when a context is defined for the IPv6

destination address, the 6LBR can infer the elided destination prefix by using the context.

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

3.2.4.2. Example of registration of multiple-addresses

As described above, a 6LN can register multiple non-link-local addresses that map to a same compression context. From the multiple address registered, only the latest address can be fully elided (SAM=11, DAM=11), and the IIDs of previously registered addresses have to be transmitted fully in-line (SAM=01, DAM=01) or in the best case can be partially elided (SAM=10, DAM=10). This is illustrated in an example below.

1) A 6LN registers first address 2001:db8::1111:2222:3333:4444 to a 6LBR. At this point the address can be fully elided using SAC=1/SAM=11 or DAC=1/DAM=11.

2) The 6LN registers second address 2001:db8::1111:2222:3333:5555 to the 6LBR. As the second address is now the latest registered, it can be fully elided using SAC=1/SAM=11 or DAC=1/DAM=11. The first address can now be partially elided using SAC=1/SAM=10 or DAC=1/DAM=10, as the first 112 bits of the address are the same between the first and the second registered addresses.

3) Expiration of registration time for the first or the second address has no impact on the compression. Hence even if the most recently registered address expires, the first address can only be partially elided (SAC=1/SAM=10, DAC=1/DAM=10). The 6LN can register a new address, or re-register an expired address, to become able to again fully elide an address.

3.2.5. Unicast and Multicast address mapping

The Bluetooth LE link layer does not support multicast. Hence traffic is always unicast between two Bluetooth LE nodes. Even in the case where a 6LBR is attached to multiple 6LNs, the 6LBR cannot do a multicast to all the connected 6LNs. If the 6LBR needs to send a multicast packet to all its 6LNs, it has to replicate the packet and unicast it on each link. However, this may not be energy-efficient and particular care must be taken if the central is battery-powered. To further conserve power, the 6LBR MUST keep track of multicast listeners at Bluetooth LE link level granularity (not at subnet granularity) and it MUST NOT forward multicast packets to 6LNs that have not registered as listeners for multicast groups the packets belong to. In the opposite direction, a 6LN always has to send packets to or through 6LBR. Hence, when a 6LN needs to transmit an IPv6 multicast packet, the 6LN will unicast the corresponding Bluetooth LE packet to the 6LBR.

4. IANA Considerations

There are no IANA considerations related to this document.

5. Security Considerations

The transmission of IPv6 over Bluetooth LE links has similar requirements and concerns for security as for IEEE 802.15.4. Bluetooth LE Link Layer security considerations are covered by the IPSP [IPSP].

Bluetooth LE Link Layer supports encryption and authentication by using the Counter with CBC-MAC (CCM) mechanism [RFC3610] and a 128-bit AES block cipher. Upper layer security mechanisms may exploit this functionality when it is available. (Note: CCM does not consume octets from the maximum per-packet L2CAP data size, since the link layer data unit has a specific field for them when they are used.)

Key management in Bluetooth LE is provided by the Security Manager Protocol (SMP), as defined in [BTCorev4.1].

The Direct Test Mode offers two setup alternatives: with and without accessible HCI. In designs with accessible HCI, the so called upper tester communicates through the HCI (which may be supported by Universal Asynchronous Receiver Transmitter (UART), Universal Serial Bus (USB) and Secure Digital transports), with the Physical and Link Layers of the Bluetooth LE device under test. In designs without accessible HCI, the upper tester communicates with the device under test through a two-wire UART interface. The Bluetooth specification

does not provide security mechanisms for the communication between the upper tester and the device under test in either case. Nevertheless, an attacker needs to physically connect a device (via one of the wired HCI types) to the device under test to be able to interact with the latter.

The IPv6 link-local address configuration described in Section 3.2.2 only reveals information about the 6LN to the 6LBR that the 6LBR already knows from the link layer connection. This means that a device using Bluetooth privacy features reveals the same information in its IPv6 link-local addresses as in its device addresses. Respectively, device not using privacy at Bluetooth level will not have privacy at IPv6 link-local address either. For non-link local addresses implementations have a choice to support, for example, [I-D.ietf-6man-default-iids], [RFC3315], [RFC3972], [RFC4941], [RFC5535], or [RFC7217].

A malicious 6LN may attempt to perform a denial of service attack on the Bluetooth LE network, for example, by flooding packets. This sort of attack is mitigated by the fact that link-local multicast is not bridged between Bluetooth LE links and by 6LBR being able to rate limit packets sent by each 6LN by making smart use of Bluetooth LE L2CAP credit-based flow control mechanism.

6. Additional contributors

Kanji Kerai, Jari Mutikainen, David Canfeng-Chen and Minjun Xi from Nokia have contributed significantly to this document.

7. Acknowledgements

The Bluetooth, Bluetooth Smart and Bluetooth Smart Ready marks are registered trademarks owned by Bluetooth SIG, Inc.

Carsten Bormann, Samita Chakrabarti, Niclas Comstedt, Alissa Cooper, Elwyn Davies, Brian Haberman, Marcel De Kogel, Jouni Korhonen, Chris Lonvick, Erik Nordmark, Erik Rivard, Dave Thaler, Pascal Thubert, Xavi Vilajosana and Victor Zhodzishsky have provided valuable feedback for this draft.

Authors would like to give special acknowledgements for Krishna Shingala, Frank Berntsen, and Bluetooth SIG's Internet Working Group for providing significant feedback and improvement proposals for this document.

8. References

8.1. Normative References

- [BTCorev4.1] Bluetooth Special Interest Group, "Bluetooth Core Specification Version 4.1", December 2013, <<https://www.bluetooth.org/en-us/specification/adopted-specifications>>.
- [IPSP] Bluetooth Special Interest Group, "Bluetooth Internet Protocol Support Profile Specification Version 1.0.0", December 2014, <<https://www.bluetooth.org/en-us/specification/adopted-specifications>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<http://www.rfc-editor.org/info/rfc4862>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<http://www.rfc-editor.org/info/rfc6282>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<http://www.rfc-editor.org/info/rfc6775>>.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <<http://www.rfc-editor.org/info/rfc7136>>.

8.2. Informative References

- [fifteendotfour]
IEEE Computer Society, "IEEE Std. 802.15.4-2011 IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)", June 2011.
- [I-D.ietf-6man-default-iids]
Gont, F., Cooper, A., Thaler, D., and S. LIU, "Recommendation on Stable IPv6 Interface Identifiers", draft-ietf-6man-default-iids-05 (work in progress), July 2015.
- [IEEE802-2001]
Institute of Electrical and Electronics Engineers (IEEE), "IEEE 802-2001 Standard for Local and Metropolitan Area Networks: Overview and Architecture", 2002.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.
- [RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610, September 2003, <<http://www.rfc-editor.org/info/rfc3610>>.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<http://www.rfc-editor.org/info/rfc3633>>.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <<http://www.rfc-editor.org/info/rfc3972>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<http://www.rfc-editor.org/info/rfc4193>>.
- [RFC4903] Thaler, D., "Multi-Link Subnet Issues", RFC 4903, DOI 10.17487/RFC4903, June 2007, <<http://www.rfc-editor.org/info/rfc4903>>.

- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<http://www.rfc-editor.org/info/rfc4944>>.
- [RFC5535] Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535, DOI 10.17487/RFC5535, June 2009, <<http://www.rfc-editor.org/info/rfc5535>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.

Authors' Addresses

Johanna Nieminen
Nokia

Email: johannamaria.nieminen@gmail.com

Teemu Savolainen
Nokia
Visiokatu 3
Tampere 33720
Finland

Email: teemu.savolainen@nokia.com

Markus Isomaki
Nokia
Otaniementie 19
Espoo 02150
Finland

Email: markus.isomaki@nokia.com

Basavaraj Patil
AT&T
1410 E. Renner Road
Richardson, TX 75082
USA

Email: basavaraj.patil@att.com

Zach Shelby
Arm
Hallituskatu 13-17D
Oulu 90100
Finland

Email: zach.shelby@arm.com

Carles Gomez
Universitat Politecnica de Catalunya/i2CAT
C/Esteve Terradas, 7
Castelldefels 08860
Spain

Email: carlesgo@entel.upc.edu

6Lo Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2015

C. Bormann
Universitaet Bremen TZI
September 19, 2014

6LoWPAN Generic Compression of Headers and Header-like Payloads (GHC)
draft-ietf-6lo-ghc-05

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. The Header Compression Coupling Problem	2
1.2. Compression Approach	3
1.3. Terminology	3
1.4. Notation	4
2. 6LoWPAN-GHC	5
3. Integrating 6LoWPAN-GHC into 6LoWPAN-HC	6
3.1. Compressing payloads (UDP and ICMPv6)	6
3.2. Compressing extension headers	6
3.3. Indicating GHC capability	7
3.4. Using the 6CIO Option	8
4. IANA considerations	9
5. Security considerations	10
6. Acknowledgements	11
7. References	13
7.1. Normative References	13
7.2. Informative References	13
Appendix A. Examples	14
Author's Address	24

1. Introduction

1.1. The Header Compression Coupling Problem

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

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

1.2. Compression Approach

The basic approach of GHC's compression function is to define a bytecode for LZ77-style compression [LZ77]. The bytecode is a series of simple instructions for the decompressor to reconstitute the uncompressed payload. These instructions include:

- o appending bytes to the reconstituted payload that are literally given with the instruction in the compressed data
- o appending a given number of zero bytes to the reconstituted payload
- o appending bytes to the reconstituted payload by copying a contiguous sequence from the payload being reconstituted ("backreferencing")
- o an ancillary instruction for setting up parameters for the backreferencing instruction in "decompression variables"
- o a stop code (optional, see Section 3.2)

The buffer for the reconstituted payload ("destination buffer") is prefixed by a predefined dictionary that can be used in the backreferencing as if it were a prefix of the payload. This predefined dictionary is built from the IPv6 addresses of the packet being reconstituted, followed by a static component, the "static dictionary".

As usual, this specification defines the decompressor operation in detail, but leaves the detailed operation of the compressor open to implementation. The compressor can be implemented as with a classical LZ77 compressor, or it can be a simple protocol encoder that just makes use of known compression opportunities.

1.3. Terminology

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

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

Terms from [RFC7228] are used in Section 5.

1.4. Notation

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

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

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

Example: The code byte template

101nssss

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

0b0ssss000

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

2. 6LoWPAN-GHC

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

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

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

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

Table 1: Bytecodes for generic header compression

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

For the purposes of the backreferences, the expansion buffer is initialized with a predefined dictionary, at the end of which the reconstituted payload begins. This dictionary is composed of the source and destination IPv6 addresses of the packet being reconstituted, followed by a 16-byte static dictionary (Figure 1).

These 48 dictionary bytes are therefore available for backreferencing, but not copied into the final reconstituted payload.

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

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

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

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

3.1. Compressing payloads (UDP and ICMPv6)

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

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

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

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

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

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

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

3.2. Compressing extension headers

Compression of specific extension headers is added in a similar way (Figure 4) (however, probably only EID 0 to 3 need to be assigned). As there is no easy way to extract the length field from the GHC-

encoded header before decoding, this would make detecting the end of the extension header somewhat complex. The easiest (and most efficient) approach is to completely elide the length field (in the same way NHC already elides the next header field in certain cases) and reconstruct it only on decompression. To serve as a terminator for the extension header, the reserved bytecode 0b10010000 has been assigned as a stop marker. Note that the stop marker is only needed for extension headers, not for the final payloads discussed in the previous subsection, the decompression of which is automatically stopped by the end of the packet.

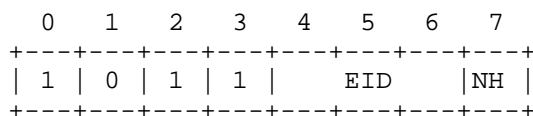


Figure 4: NHC byte for extension header GHC

3.3. Indicating GHC capability

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

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

To minimize the impact on [RFC6775], we define an ND option 6LoWPAN Capability Indication (6CIO), as illustrated in Figure 5. (For the fields marked by an underscore in Figure 5, see Section 3.4.)

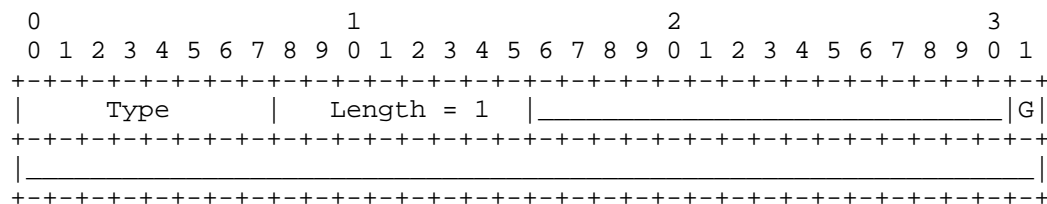


Figure 5: 6LoWPAN Capability Indication Option (6CIO)

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

Once a node receives either an explicit or an implicit indication of GHC capability from another node, it may send GHC-compressed packets to that node. Where that capability has not been recently confirmed, similar to the way PLPMTUD [RFC4821] finds out about changes in the network, a node SHOULD make use of NUD (neighbor unreachability detection) failures to switch back to basic 6LoWPAN header compression [RFC6282].

3.4. Using the 6CIO Option

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

6CIO can also be used for future options that need to be negotiated between 6LoWPAN peers; an IANA registry is used to assign the flags. Bits marked by underscores in Figure 5 are unassigned and available for future assignment. They MUST be sent as zero and MUST be ignored on reception until assigned by IANA. Length values larger than 1 MUST be accepted by implementations in order to enable future extensions; the additional bits in the option are then deemed unassigned in the same way. For the purposes of the IANA registry, the bits are numbered in most-significant-bit-first order from the 16th bit of the option onward: the 16th bit is flag number 0, the 31st bit (the G bit) is flag number 15, up to the 63rd bit for flag number 47. (Additional bits may also be used by a follow-on version of this document if some bit combinations that have been left unassigned here are then used in an upward compatible manner.)

Flag numbers 0 to 7 are reserved for experiments. They MUST NOT be used for actual deployments.

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

- o The option can be used in any ND packet.
- o Specific bits are set in the option to indicate that a capability is present in the sender. (There may be other ways to infer this information, as is the case in this specification.) Bit combinations may be used as desired. The absence of the capability `_indication_` is signaled by setting these bits to zero; this does not necessarily mean that the capability is absent.

- o The intention is not to modify the semantics of the specific ND packet carrying the option, but to provide the general capability indication described above.
- o Specifications have to be designed such that receivers that do not receive or do not process such a capability indication can still interoperate (presumably without exploiting the indicated capability).
- o The option is meant to be used sparsely, i.e. once a sender has reason to believe the capability indication has been received, there no longer is a need to continue sending it.

4. IANA considerations

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

In the IANA registry for the "LOWPAN_NHC Header Type" (in the "IPv6 Low Power Personal Area Network Parameters"), IANA is requested to add the assignments in Figure 6.

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

Figure 6: IANA assignments for the NHC byte

IANA is requested to allocate an ND option number for the "6LoWPAN Capability Indication Option (6CIO)" ND option format in the Registry "IPv6 Neighbor Discovery Option Formats" [RFC4861].

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

0..7: reserved for experiments	[RFCthis]
8..14: unassigned	
15: GHC capable bit (G bit)	[RFCthis]
16..47: unassigned	

Figure 7: IANA assignments for the 6LoWPAN capability bits

5. Security considerations

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

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

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

As with any LZ77 scheme, decompression bombs (compressed packets crafted to expand so much that the decompressor is overloaded) are a problem. An attacker cannot send a GHC decompressor into a tight loop for too long, because the MTU will be reached quickly. Some amplification of an attack from inside the compressed link is possible, though. Using a constrained node in a constrained network as a DoS attack source is usually not very useful, though, except maybe against other nodes in that constrained network. The worst case for an attack to the outside is a not-so-constrained device using a (typically not-so-constrained) edge router to attack by forwarding out of its Ethernet interface. The worst-case amplification of GHC is 17, so an MTU-size packet can be generated from a 6LoWPAN packet of 76 bytes. The 6LoWPAN network is still constrained, so the amplification at the edge router turns an entire 250 kbit/s 802.15.4 network (assuming a theoretical upper bound of 225 kbit/s throughput to a single-hop adjacent node) into a 3.8 Mbit/s attacker.

The amplification may be more important inside the 6LoWPAN, if there is a way to obtain reflection (otherwise the packet is likely to simply stay compressed on the way and do little damage), e.g., by pinging a node using a decompression bomb, somehow keeping that node from re-compressing the ping response (which would probably require something more complex than simple runs of zeroes, so the worst-case

amplification is likely closer to 9). Or, if there are nodes that do not support GHC, those can be attacked via a router that is then forced to decompress.

All these attacks are mitigated by some form of network access control.

In a 6LoWPAN stack, sensitive information will normally be protected by transport or application (or even IP) layer security, which are all above the adaptation layer, leaving no sensitive information to compress at the GHC level. However, a 6LoWPAN deployment that entirely depends on MAC layer security may be vulnerable to attacks that exploit redundancy information disclosed by compression to recover information about secret values. The attacker would need to be in radio range to observe the compressed packets. Since compression is stateless, the attacker would need to entice the party sending the secret value to also send some value controlled (or at least usefully varying and knowable) by the attacker in (what becomes the first adaptation layer fragment of) the same packet. This attack is fully mitigated by not exposing secret values to the adaptation layer, or by not using GHC in deployments where this is done.

6. Acknowledgements

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

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

Using these pcap files as a corpus, the static dictionary was developed, and the bit allocations validated, based on research by Sebastian Dominik.

Erik Nordmark provided input that helped shaping the 6CIO option. Thomas Bjorklund proposed simplifying the predefined dictionary.

Yoshihiro Ohba insisted on clarifying the notation used for the definition of the bytecodes and their bitfields. Ulrich Herberg provided some additional review and suggested expanding the introductory material, and with Hannes Tschofenig and Brian Haberman

he helped come up with the IANA policy for the "6LoWPAN capability bits" assignments in the 6CIO option.

The IESG reviewers Richard Barnes and Stephen Farrell have contributed issues to the security considerations section; they and Barry Leiba, as well as GEN-ART reviewer Vijay K. Gurbani also have provided editorial improvements.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

7.2. Informative References

- [I-D.bormann-6lo-6lowpan-roadmap] Bormann, C., "6LoWPAN Roadmap and Implementation Guide", draft-bormann-6lo-6lowpan-roadmap-00 (work in progress), October 2013.
- [I-D.oflynn-6lowpan-icmphc] O'Flynn, C., "ICMPv6/ND Compression for 6LoWPAN Networks", draft-oflynn-6lowpan-icmphc-00 (work in progress), July 2010.
- [LZ77] Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343, May 1977.
- [RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

- [RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, March 2010.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

Appendix A. Examples

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

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

```

IP header:
 60 00 00 00 00 08 3a ff fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 20 24 ff 02 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 1a
Payload:
 9b 00 6b de 00 00 00 00
Dictionary:
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 20 24
 ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
 16 fe fd 17 fe fd 00 01 00 00 00 00 00 01 00 00
copy: 04 9b 00 6b de
4 nulls: 82
Compressed:
 04 9b 00 6b de 82
Was 8 bytes; compressed to 6 bytes, compression factor 1.33

```

Figure 8: A simple RPL example

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

```

IP header:
 60 00 00 00 00 5c 3a ff fe 80 00 00 00 00 00 00
 02 1c da ff fe 00 30 23 ff 02 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 1a
Payload:
 9b 01 7a 5f 00 f0 01 00 88 00 00 00 20 02 0d b8
 00 00 00 00 00 00 00 00 ff fe 00 fa ce 04 0e 00 14
 09 ff 00 00 01 00 00 00 00 00 00 00 08 1e 80 20
 ff ff ff ff ff ff ff ff 00 00 00 00 20 02 0d b8
 00 00 00 00 00 00 00 00 ff fe 00 fa ce 03 0e 40 00
 ff ff ff ff 20 02 0d b8 00 00 00 00
Dictionary:
 fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23
 ff 02 00 00 00 00 00 00 00 00 00 00 00 00 00 1a
 16 fe fd 17 fe fd 00 01 00 00 00 00 00 01 00 00
copy: 06 9b 01 7a 5f 00 f0
ref(9): 01 00 -> ref 1lnnnkkk 0 7: c7
copy: 01 88
3 nulls: 81
copy: 04 20 02 0d b8
7 nulls: 85
ref(60): ff fe 00 -> ref 10lnssss 0 7/1lnnnkkk 1 1: a7 c9
copy: 08 fa ce 04 0e 00 14 09 ff
ref(39): 00 00 01 00 00 -> ref 10lnssss 0 4/1lnnnkkk 3 2: a4 da
5 nulls: 83
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 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:
 06 9b 01 7a 5f 00 f0 c7 01 88 81 04 20 02 0d b8
 85 a7 c9 08 fa ce 04 0e 00 14 09 ff a4 da 83 06
 08 1e 80 20 ff ff c0 d0 82 b4 f0 03 03 0e 40 c7
 a3 c9 a2 f0
Was 92 bytes; compressed to 52 bytes, compression factor 1.77

```

Figure 9: A longer RPL example

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

IP header:

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

Payload:

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

Dictionary:

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

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

ref(60): 20 02 0d b8 00 00 00 00 00 00 00 00 ff fe 00 33 44

-> ref 10lnssss 1 5/1lnnnkkk 6 4: b5 f4

copy: 08 06 14 00 80 f1 00 fe 80

9 nulls: 87

ref(66): ff fe 00 11 22 -> ref 10lnssss 0 7/1lnnnkkk 3 5: a7 dd

Compressed:

```
0c 9b 02 58 7d 01 80 00 f1 05 12 00 80 b5 f4 08
06 14 00 80 f1 00 fe 80 87 a7 dd
```

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

Figure 10: An RPL DAO message

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

IP header:

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

Payload:

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

Dictionary:

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

copy: 04 87 00 a7 68

4 nulls: 82

ref(40): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23

-> ref 10lnssss 1 3/1lnnnkkk 6 0: b3 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 b3 f0 04 01 01 3b d3 82 02 1f
02 83 02 06 00 a2 db 02 20 24
```

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

Figure 11: An ND neighbor solicitation

Figure 12 shows the compression of an ND neighbor advertisement.

IP header:

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

Payload:

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

Dictionary:

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

copy: 05 88 00 26 6c c0

3 nulls: 81

ref(56): fe 80 00 00 00 00 00 00 02 1c da ff fe 00 30 23

-> ref 10lnssss 1 5/1lnnnkkk 6 0: b5 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 b5 f0 04 02 01 fa ce 82 02
1f 02 83 02 06 00 a2 db 02 20 24
```

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

Figure 12: An ND neighbor advertisement

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

IP header:

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

Payload:

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

Dictionary:

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

copy: 04 85 00 90 65

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

copy: 02 02 ac

ref(50): de 48 00 00 00 00 01

-> ref 10lnssss 0 5/11nnnkkk 5 3: a5 eb

6 nulls: 84

Compressed:

```
04 85 00 90 65 de 02 02 ac a5 eb 84
```

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

Figure 13: An ND router solicitation

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


```

IP header:
 60 00 00 00 00 60 3a ff fe 80 00 00 00 00 00 00
 10 34 00 ff fe 00 11 22 fe 80 00 00 00 00 00 00
 ae de 48 00 00 00 00 01
Payload:
 86 00 55 c9 40 00 0f a0 1c 5a 38 17 00 00 07 d0
 01 01 11 22 00 00 00 00 03 04 40 40 ff ff ff ff
 ff ff ff ff 00 00 00 00 20 02 0d b8 00 00 00 00
 00 00 00 00 00 00 00 00 20 02 40 10 00 00 03 e8
 20 02 0d b8 00 00 00 00 21 03 00 01 00 00 00 00
 20 02 0d b8 00 00 00 00 00 00 00 00 ff fe 00 11 22
Dictionary:
 fe 80 00 00 00 00 00 00 10 34 00 ff fe 00 11 22
 fe 80 00 00 00 00 00 00 ae de 48 00 00 00 00 01
 16 fe fd 17 fe fd 00 01 00 00 00 00 00 01 00 00
copy: 0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17
2 nulls: 80
copy: 06 07 d0 01 01 11 22
4 nulls: 82
copy: 06 03 04 40 40 ff ff
ref(2): ff ff -> ref 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 00
-> ref 10lnssss 0 9/11nnnkkk 4 6: a9 e6
ref(40): 20 02 0d b8 00 00 00 00 00 00 00
-> ref 10lnssss 1 3/11nnnkkk 1 5: b3 cd
ref(128): ff fe 00 11 22
-> ref 10lnssss 0 15/11nnnkkk 3 3: af db
Compressed:
0c 86 00 55 c9 40 00 0f a0 1c 5a 38 17 80 06 07
d0 01 01 11 22 82 06 03 04 40 40 ff ff c0 d0 82
04 20 02 0d b8 8a 04 20 02 40 10 a4 cb 01 e8 a2
f0 02 21 03 a9 e6 b3 cd af db
Was 96 bytes; compressed to 58 bytes, compression factor 1.66

```

Figure 14: An ND router advertisement

Figure 15 shows the compression of a DTLS application data packet with a net payload of 13 bytes of cleartext, and 8 bytes of

authenticator (note that the IP header is not relevant for this example and has been set to 0). This makes good use of the static dictionary, and is quite effective crunching out the redundancy in the TLS_PSK_WITH_AES_128_CCM_8 header, leading to a net reduction by 15 bytes.

IP header:

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

Payload:

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

Dictionary:

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

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

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

copy: 01 1d

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

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

copy: b5 d4 22 d4 ed 2b

Compressed:

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

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

Figure 15: A DTLS application data packet

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

IP header:

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

Payload:

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

Dictionary:

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

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

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

copy: 03 05 00 16

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

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

Compressed:

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

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

Figure 16: Another DTLS application data packet

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

IP header:

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

Payload:

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

Dictionary:

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

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

9 nulls: 87

copy: 01 36

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

copy: 01 2a

7 nulls: 85

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

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

copy: 9f 20 92 92

3 nulls: 81

copy: 05 02 c0 a8 01 00

Compressed:

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

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

Figure 17: A DTLS handshake packet (client hello)

Author's Address

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

Phone: +49-421-218-63921
Email: carbo@tzi.org

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: March 9, 2015

J. Schoenwaelder
A. Sehgal
Jacobs University
T. Tsou
Huawei Technologies (USA)
C. Zhou
Huawei Technologies
September 5, 2014

Definition of Managed Objects for IPv6 over Low-Power Wireless Personal
Area Networks (6LoWPANs)
draft-ietf-6lo-lowpan-mib-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 9, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. The Internet-Standard Management Framework	2
3. Conventions	3
4. Overview	3
5. Relationship to Other MIB Modules	7
6. Definitions	7
7. Security Considerations	24
8. IANA Considerations	25
9. Acknowledgements	25
10. References	25
10.1. Normative References	25
10.2. Informative References	26

1. Introduction

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

While a MIB module provides a direct binding for accessing data via the Simple Network Management Protocol (SNMP) [RFC3410], supporting SNMP may not always be affordable on constrained devices. Other protocols to access data modeled in MIB modules are possible and proposals have been made recently to provide bindings to the Constrained Application Protocol (CoAP) [RFC7252].

2. The Internet-Standard Management Framework

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

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

3. Conventions

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

4. Overview

The left part of Figure 1 provides an overview of the IETF protocols designed for constrained devices. The right part lists the MIB modules providing monitoring and troubleshooting support ([RFC4113], [RFC4292], [RFC4293], [RFC2863]). The LOWPAN-MIB defined in this document fills a hole by providing monitoring and troubleshooting support for the 6LoWPAN layer.

Protocol Layer	MIB Modules
+-----+	+-----+
CoAP [RFC7252]	UDP-MIB [RFC4113]
+-----+	+-----+
UDP [RFC0768]	IP-MIB [RFC4293]
+-----+	+-----+
IPv6 [RFC2460]	IP-FORWARD-MIB [RFC4292]
ICMPv6 [RFC4443]	+-----+
+-----+	LOWPAN-MIB [RFCXXXX]
6LoWPAN [RFC4944]	+-----+
+-----+	IF-MIB [RFC2863]
IEEE 802.15.4, ...	+-----+
+-----+	

/* RFC Ed.: replace XXXX above with RFC number
and remove this note */

Figure 1: Protocol Layers and MIB Modules

The LOWPAN-MIB module is primarily a collection of counters that reflect how 6LoWPAN datagrams are processed by the 6LoWPAN layer. The objects are defined twice, once to report the global statistics as seen by the 6LoWPAN layer and once to report per interface 6LoWPAN layer statistics. The per interface statistics are optional to implement. The object identifier registration tree has the following structure:

/* RFC Ed.: replace XXXX below with IANA assigned OID number
and remove this note */


```

---- lowpanMIB(1.3.6.1.2.1.XXXX)
+---- lowpanNotifications(0)
+---- lowpanObjects(1)
|   +---- lowpanStats(1)
|   |   +--r- lowpanReasmTimeout(1)           Unsigned32
|   |   +--r- lowpanInReceives(2)           Counter32
|   |   +--r- lowpanInHdrErrors(3)          Counter32
|   |   +--r- lowpanInMeshReceives(4)       Counter32
|   |   +--r- lowpanInMeshForwds(5)         Counter32
|   |   +--r- lowpanInMeshDelivers(6)       Counter32
|   |   +--r- lowpanInReasmReqds(7)         Counter32
|   |   +--r- lowpanInReasmFails(8)         Counter32
|   |   +--r- lowpanInReasmOKs(9)          Counter32
|   |   +--r- lowpanInCompReqds(10)         Counter32
|   |   +--r- lowpanInCompFails(11)         Counter32
|   |   +--r- lowpanInCompOKs(12)          Counter32
|   |   +--r- lowpanInDiscards(13)         Counter32
|   |   +--r- lowpanInDelivers(14)         Counter32
|   |   +--r- lowpanOutRequests(15)         Counter32
|   |   +--r- lowpanOutCompReqds(16)        Counter32
|   |   +--r- lowpanOutCompFails(17)        Counter32
|   |   +--r- lowpanOutCompOKs(18)         Counter32
|   |   +--r- lowpanOutFragReqds(19)        Counter32
|   |   +--r- lowpanOutFragFails(20)        Counter32
|   |   +--r- lowpanOutFragOKs(21)         Counter32
|   |   +--r- lowpanOutFragCreates(22)      Counter32
|   |   +--r- lowpanOutMeshHopLimitExceeds(23) Counter32
|   |   +--r- lowpanOutMeshNoRoutes(24)     Counter32
|   |   +--r- lowpanOutMeshRequests(25)     Counter32
|   |   +--r- lowpanOutMeshForwds(26)      Counter32
|   |   +--r- lowpanOutMeshTransmits(27)    Counter32
|   |   +--r- lowpanOutDiscards(28)        Counter32
|   |   +--r- lowpanOutTransmits(29)       Counter32
|   +---- lowpanIfStatsTable(2)
|   |   +---- lowpanIfStatsEntry(1) [ifIndex]
|   |   |   +--r- lowpanIfReasmTimeout(1)   Unsigned32
|   |   |   +--r- lowpanIfInReceives(2)     Counter32
|   |   |   +--r- lowpanIfInHdrErrors(3)     Counter32
|   |   |   +--r- lowpanIfInMeshReceives(4)  Counter32
|   |   |   +--r- lowpanIfInMeshForwds(5)    Counter32
|   |   |   +--r- lowpanIfInMeshDelivers(6)  Counter32
|   |   |   +--r- lowpanIfInReasmReqds(7)    Counter32
|   |   |   +--r- lowpanIfInReasmFails(8)    Counter32
|   |   |   +--r- lowpanIfInReasmOKs(9)      Counter32
|   |   |   +--r- lowpanIfInCompReqds(10)    Counter32
|   |   |   +--r- lowpanIfInCompFails(11)    Counter32
|   |   |   +--r- lowpanIfInCompOKs(12)     Counter32
|   |   |   +--r- lowpanIfInDiscards(13)    Counter32

```

```

|      +--r- lowpanIfInDelivers(14)           Counter32
|      +--r- lowpanIfOutRequests(15)          Counter32
|      +--r- lowpanIfOutCompReqds(16)          Counter32
|      +--r- lowpanIfOutCompFails(17)          Counter32
|      +--r- lowpanIfOutCompOKs(18)            Counter32
|      +--r- lowpanIfOutFragReqds(19)          Counter32
|      +--r- lowpanIfOutFragFails(20)          Counter32
|      +--r- lowpanIfOutFragOKs(21)            Counter32
|      +--r- lowpanIfOutFragCreates(22)         Counter32
|      +--r- lowpanIfOutMeshHopLimitExceeds(23) Counter32
|      +--r- lowpanIfOutMeshNoRoutes(24)        Counter32
|      +--r- lowpanIfOutMeshRequests(25)        Counter32
|      +--r- lowpanIfOutMeshForwds(26)          Counter32
|      +--r- lowpanIfOutMeshTransmits(27)       Counter32
|      +--r- lowpanIfOutDiscards(28)            Counter32
|      +--r- lowpanIfOutTransmits(29)           Counter32
+----- lowpanConformance(2)
+----- lowpanGroups(1)
|   +----- lowpanStatsGroup(1)
|   +----- lowpanStatsMeshGroup(2)
|   +----- lowpanIfStatsGroup(3)
|   +----- lowpanIfStatsMeshGroup(4)
+----- lowpanCompliances(2)
+----- lowpanCompliance(1)

```

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

The Case Diagram [CASE] in Figure 2 illustrates the conceptual relationships between the counters. Implementations may choose to implement the processing of 6LoWPAN datagrams in a different order.

The generic InDiscards and OutDiscards counters can be incremented anytime when 6LoWPAN datagrams are discarded due to reasons not covered by the other more specific counters. For example, an implementation discarding 6LoWPAN datagrams while all buffers are used for ongoing packet reassemblies will increment the relevant InDiscards counters for each discarded 6LoWPAN datagram.

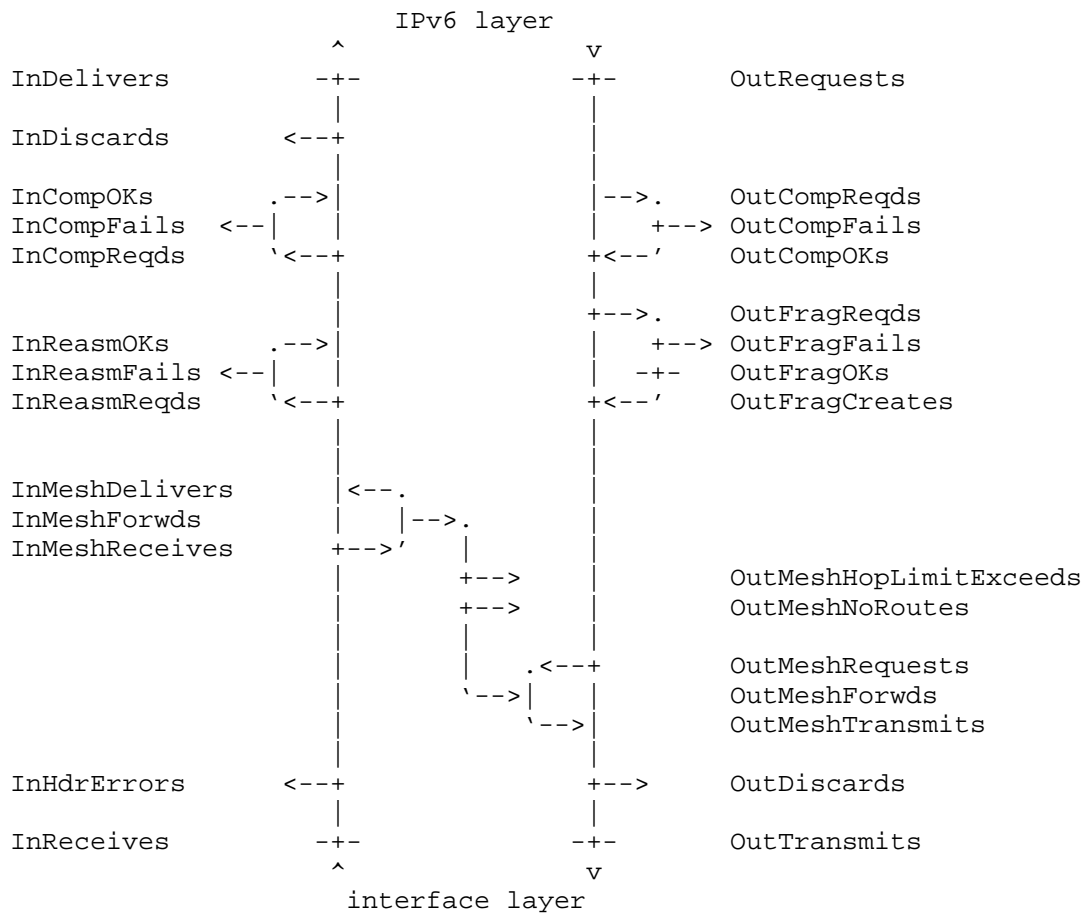


Figure 2: Conceptual Relationship between LOWPAN-MIB Counters

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

The compression related counters provide insights into compression requests and in particular also compression related failures. Note that the diagram is conceptual in the sense that compression happens after reassembly for incoming 6LoWPAN datagrams and compression happens before fragmentation for outgoing 6LoWPAN datagrams. Implementations may choose to implement things slightly differently. For example, implementations may decompress FRAG1 fragments as soon as they are received, not waiting for reassembly to complete.

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

5. Relationship to Other MIB Modules

The MIB module imports definitions from SNMPv2-SMI [RFC2578], SNMPv2-CONF [RFC2580], and IF-MIB [RFC2863].

6. Definitions

```
LOWPAN-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    MODULE-IDENTITY, OBJECT-TYPE, Unsigned32, Counter32, mib-2
        FROM SNMPv2-SMI                                -- RFC 2578
    OBJECT-GROUP, MODULE-COMPLIANCE
        FROM SNMPv2-CONF                                -- RFC 2580
    ifIndex FROM IF-MIB;                                -- RFC 2863
```

```
lowpanMIB      MODULE-IDENTITY
```

```
    LAST-UPDATED      "201409050000Z"
```

```
    ORGANIZATION
```

```
        "IETF IPv6 over Networks of Resource-constrained Nodes
         Working Group"
```

```
    CONTACT-INFO
```

```
        "WG Email: 6lo@ietf.org
```

```
        WG Web:   http://tools.ietf.org/wg/6lo/
```

```
        Juergen Schoenwaelder
```

```
        Jacobs University Bremen
```

```
        Email: j.schoenwaelder@jacobs-university.de
```

```
        Anuj Sehgal
```

```
        Jacobs University Bremen
```

```
        Email: s.anuj@jacobs-university.de
```

```
        Tina Tsou
```

```
        Huawei Technologies
```

```
        Email: tina.tsou.zouting@huawei.com
```

```
        Cathy Zhou
```

```
        Huawei Technologies
```

```
        Email: cathyzhou@huawei.com"
```

DESCRIPTION

"The MIB module for monitoring nodes implementing the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) protocol.

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

REVISION "201409050000Z"

DESCRIPTION

"Initial version, published as RFC XXXX."

-- RFC Ed.: replace XXXX with RFC number and remove this note

::= { mib-2 YYYY }

-- RFC Ed.: replace YYYY with IANA assigned number

-- object definitions

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

lowpanStats OBJECT IDENTIFIER ::= { lowpanObjects 1 }

lowpanReasmTimeout OBJECT-TYPE

SYNTAX Unsigned32

UNITS "seconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The maximum number of seconds that received fragments are held while they are awaiting reassembly at this entity."

::= { lowpanStats 1 }

lowpanInReceives OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of 6LoWPAN datagrams received, including those received in error."

```
 ::= { lowpanStats 2 }

lowpanInHdrErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of received 6LoWPAN datagrams discarded due to
         errors in their headers, including unknown dispatch values."
    ::= { lowpanStats 3 }

lowpanInMeshReceives OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of received 6LoWPAN datagrams with a MESH
         header."
    ::= { lowpanStats 4 }

lowpanInMeshForwds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of received 6LoWPAN datagrams requiring MESH
         forwarding."
    ::= { lowpanStats 5 }

lowpanInMeshDelivers OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of received 6LoWPAN datagrams with a MESH header
         delivered to the local system."
    ::= { lowpanStats 6 }

lowpanInReasmReqds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of received 6LoWPAN fragments that needed to
         be reassembled. This includes both FRAG1 and FRAGN 6LoWPAN
         datagrams."
    ::= { lowpanStats 7 }
```

lowpanInReasmFails OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

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

::= { lowpanStats 8 }

lowpanInReasmOKs OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of IPv6 packets successfully reassembled."

::= { lowpanStats 9 }

lowpanInCompReqds OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of 6LoWPAN datagrams requiring header decompression."

::= { lowpanStats 10 }

lowpanInCompFails OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of 6LoWPAN datagrams where header decompression failed (e.g., because the necessary context information was not available)."

::= { lowpanStats 11 }

lowpanInCompOKs OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of 6LoWPAN datagrams where header decompression was successful."

::= { lowpanStats 12 }

lowpanInDiscards OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "The number of received 6LoWPAN datagrams for which no
 problems were encountered to prevent their continued
 processing, but were discarded (e.g., for lack of buffer
 space). Note that this counter does not include any
 datagrams discarded due to a reassembly failure or a
 compression failure."
 ::= { lowpanStats 13 }

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

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

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

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


```
lowpanOutCompOKs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The total number of IPv6 packets for which header
         compression was successful."
    ::= { lowpanStats 18 }

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

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

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

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

lowpanOutMeshHopLimitExceeds OBJECT-TYPE
    SYNTAX      Counter32
```

```
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of 6LoWPAN datagrams with a MESH header that
    were dropped because the hop limit has been exceeded."
 ::= { lowpanStats 23 }

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

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

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

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

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

DESCRIPTION

"The number of IPv6 packets for which no problem was encountered to prevent their transmission to their destination, but were discarded (e.g., for lack of buffer space)."

::= { lowpanStats 28 }

lowpanOutTransmits OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of 6LoWPAN datagram that this entity supplied to the lower layers for transmission."

::= { lowpanStats 29 }

lowpanIfStatsTable OBJECT-TYPE

SYNTAX SEQUENCE OF LowpanIfStatsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"A table providing per interface statistics."

::= { lowpanObjects 2 }

lowpanIfStatsEntry OBJECT-TYPE

SYNTAX LowpanIfStatsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry providing statistics for a specific interface."

INDEX { ifIndex }

::= { lowpanIfStatsTable 1 }

LowpanIfStatsEntry ::= SEQUENCE {

lowpanIfReasmTimeout	Unsigned32,
lowpanIfInReceives	Counter32,
lowpanIfInHdrErrors	Counter32,
lowpanIfInMeshReceives	Counter32,
lowpanIfInMeshForwds	Counter32,
lowpanIfInMeshDelivers	Counter32,
lowpanIfInReasmReqds	Counter32,
lowpanIfInReasmFails	Counter32,
lowpanIfInReasmOKs	Counter32,
lowpanIfInCompReqds	Counter32,
lowpanIfInCompFails	Counter32,
lowpanIfInCompOKs	Counter32,
lowpanIfInDiscards	Counter32,
lowpanIfInDelivers	Counter32,

```
    lowpanIfOutRequests          Counter32,
    lowpanIfOutCompReqds         Counter32,
    lowpanIfOutCompFails         Counter32,
    lowpanIfOutCompOKs           Counter32,
    lowpanIfOutFragReqds         Counter32,
    lowpanIfOutFragFails         Counter32,
    lowpanIfOutFragOKs           Counter32,
    lowpanIfOutFragCreates        Counter32,
    lowpanIfOutMeshHopLimitExceeds Counter32,
    lowpanIfOutMeshNoRoutes       Counter32,
    lowpanIfOutMeshRequests       Counter32,
    lowpanIfOutMeshForwds         Counter32,
    lowpanIfOutMeshTransmits      Counter32,
    lowpanIfOutDiscards           Counter32,
    lowpanIfOutTransmits          Counter32
}

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

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

lowpanIfInHdrErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams received on this
         interface that were discarded due to errors in
         their headers, including unknown dispatch values."
    ::= { lowpanIfStatsEntry 3 }

lowpanIfInMeshReceives OBJECT-TYPE
    SYNTAX      Counter32
```

```
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "The number of 6LoWPAN datagrams reveived on this
    interface with a MESH header."
 ::= { lowpanIfStatsEntry 4 }

lowpanIfInMeshForwds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams received on this
        interface requiring MESH forwarding."
    ::= { lowpanIfStatsEntry 5 }

lowpanIfInMeshDelivers OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams received on this
        interface with a MESH header delivered to the local
        system."
    ::= { lowpanIfStatsEntry 6 }

lowpanIfInReasmReqds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN fragments received on this
        interface that needed to be reassembled. This
        includes both FRAG1 and FRAGN 6LoWPAN datagrams."
    ::= { lowpanIfStatsEntry 7 }

lowpanIfInReasmFails OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of failures detected by the re-assembly
        algorithm (e.g., timeouts) for datagrams received
        on this interface. Note that this is not necessarily
        a count of discarded 6LoWPAN fragments since
        implementations can lose track of the number
        of fragments by combining them as received."
    ::= { lowpanIfStatsEntry 8 }
```

```
lowpanIfInReasmOKs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of IPv6 packets successfully reassembled
         from fragments received on this interface."
    ::= { lowpanIfStatsEntry 9 }

lowpanIfInCompReqds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of 6LoWPAN datagrams received on this
         interface requiring header decompression."
    ::= { lowpanIfStatsEntry 10 }

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

lowpanIfInCompOKs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of 6LoWPAN datagrams received on this
         interface where header decompression was successful."
    ::= { lowpanIfStatsEntry 12 }

lowpanIfInDiscards OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of 6LoWPAN datagrams received on this
         interface for which no problems were encountered to
         prevent their continued processing, but were discarded
         (e.g., for lack of buffer space). Note that this
         counter does not include any datagrams discarded due
```

to a reassembly failure or a compression failure."
 ::= { lowpanIfStatsEntry 13 }

lowpanIfInDelivers OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of IPv6 packets received on this interface that were successfully delivered to the IPv6 layer."

::= { lowpanIfStatsEntry 14 }

lowpanIfOutRequests OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of IPv6 packets supplied by the IPv6 layer to be sent over this interface."

::= { lowpanIfStatsEntry 15 }

lowpanIfOutCompReqds OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of IPv6 packets to be sent over this interface for which header compression was attempted."

::= { lowpanIfStatsEntry 16 }

lowpanIfOutCompFails OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of IPv6 packets to be sent over this interface for which header compression failed."

::= { lowpanIfStatsEntry 17 }

lowpanIfOutCompOKs OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of IPv6 packets to be sent over this interface for which header compression was

```
        successful."
 ::= { lowpanIfStatsEntry 18 }

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

lowpanIfOutFragFails OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of IPv6 packets to be sent over this
        interface that have been discarded because
        fragmentation failed."
 ::= { lowpanIfStatsEntry 20 }

lowpanIfOutFragOKs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of IPv6 packets to be sent over this
        interface that have been successfully fragmented."
 ::= { lowpanIfStatsEntry 21 }

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

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



```
        "The number of 6LoWPAN datagrams to be sent on this
        interface with a MESH header that were dropped
        because the hop limit has been exceeded."
 ::= { lowpanIfStatsEntry 23 }

lowpanIfOutMeshNoRoutes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams to be sent on this
        interface with a MESH header that were dropped
        because there was no forwarding information available."
 ::= { lowpanIfStatsEntry 24 }

lowpanIfOutMeshRequests OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams to be sent on this
        interface requiring MESH header encapsulation."
 ::= { lowpanIfStatsEntry 25 }

lowpanIfOutMeshForwds OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams to be sent on this
        interface with a MESH header for which suitable
        forwarding information was available."
 ::= { lowpanIfStatsEntry 26 }

lowpanIfOutMeshTransmits OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The number of 6LoWPAN datagrams to be send on this
        interface with a MESH header created."
 ::= { lowpanIfStatsEntry 27 }

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

```
        "The number of IPv6 packets to be sent over this
        interface for which no problem was encountered to
        prevent their transmission to their destination, but
        were discarded (e.g., for lack of buffer space)."
```

```
 ::= { lowpanIfStatsEntry 28 }
```

```
lowpanIfOutTransmits OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of 6LoWPAN datagrams to be sent on
        this interface that this entity supplied to the lower
        layers for transmission."
    ::= { lowpanIfStatsEntry 29 }
```

```
-- conformance definitions
```

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

```
lowpanCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "Compliance statement for systems that implement 6LoWPAN."
    MODULE      -- this module
    MANDATORY-GROUPS {
        lowpanStatsGroup
    }
    GROUP        lowpanStatsMeshGroup
    DESCRIPTION
        "This group is mandatory for implementations that process
        or forward 6LoWPAN datagrams with mesh headers."
    GROUP        lowpanIfStatsGroup
    DESCRIPTION
        "This group is mandatory for implementations that expose
        per interface statistics."
    GROUP        lowpanIfStatsMeshGroup
    DESCRIPTION
        "This group is mandatory for implementations that expose
        per interface statistics and that process or forward
        6LoWPAN datagrams with mesh headers."
    ::= { lowpanCompliances 1 }
```

```
lowpanStatsGroup OBJECT-GROUP
    OBJECTS {
        lowpanReasmTimeout,
        lowpanInReceives,
```

```

        lowpanInHdrErrors,
        lowpanInMeshReceives,
        lowpanInReasmReqds,
        lowpanInReasmFails,
        lowpanInReasmOKs,
        lowpanInCompReqds,
        lowpanInCompFails,
        lowpanInCompOKs,
        lowpanInDiscards,
        lowpanInDelivers,
        lowpanOutRequests,
        lowpanOutCompReqds,
        lowpanOutCompFails,
        lowpanOutCompOKs,
        lowpanOutFragReqds,
        lowpanOutFragFails,
        lowpanOutFragOKs,
        lowpanOutFragCreates,
        lowpanOutDiscards,
        lowpanOutTransmits
    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing information and
        statistics about the processing of 6LoWPAN datagrams,
        excluding counters covering the processing of datagrams
        with a mesh headers."
    ::= { lowpanGroups 1 }

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

lowpanIfStatsGroup OBJECT-GROUP
    OBJECTS {

```

```

        lowpanIfReasmTimeout,
        lowpanIfInReceives,
        lowpanIfInHdrErrors,
        lowpanIfInMeshReceives,
        lowpanIfInReasmReqds,
        lowpanIfInReasmFails,
        lowpanIfInReasmOKs,
        lowpanIfInCompReqds,
        lowpanIfInCompFails,
        lowpanIfInCompOKs,
        lowpanIfInDiscards,
        lowpanIfInDelivers,
        lowpanIfOutRequests,
        lowpanIfOutCompReqds,
        lowpanIfOutCompFails,
        lowpanIfOutCompOKs,
        lowpanIfOutFragReqds,
        lowpanIfOutFragFails,
        lowpanIfOutFragOKs,
        lowpanIfOutFragCreates,
        lowpanIfOutDiscards,
        lowpanIfOutTransmits
    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing per interface
        information and statistics about the processing
        of 6LoWPAN datagrams, excluding counters covering
        the processing of datagrams with a mesh headers."
    ::= { lowpanGroups 3 }

lowpanIfStatsMeshGroup OBJECT-GROUP
    OBJECTS {
        lowpanIfInMeshForwds,
        lowpanIfInMeshDelivers,
        lowpanIfOutMeshHopLimitExceeds,
        lowpanIfOutMeshNoRoutes,
        lowpanIfOutMeshRequests,
        lowpanIfOutMeshForwds,
        lowpanIfOutMeshTransmits
    }
    STATUS          current
    DESCRIPTION
        "A collection of objects providing per interface
        information and statistics about the processing
        of 6LoWPAN datagrams with a 6LoWPAN mesh header."
    ::= { lowpanGroups 4 }

```

END

7. Security Considerations

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

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

The read-only counters provide insights into the amount of 6LoWPAN traffic a node is receiving or transmitting. This might provide information whether a device is regularly exchanging information with other devices or whether a device is mostly not participating in any communication (e.g., the device might be "easier" to take away unnoticed). The reassembly counters could be used to direct denial of service attacks on the reassembly mechanism.

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

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

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

8. IANA Considerations

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

The MIB module in this document uses the following IANA-assigned OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

Descriptor	OBJECT IDENTIFIER value
-----	-----
lowpanMIB	{ mib-2 YYYY }

9. Acknowledgements

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

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

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

10.2. Informative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.
- [RFC4113] Fenner, B. and J. Flick, "Management Information Base for the User Datagram Protocol (UDP)", RFC 4113, June 2005.
- [RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292, April 2006.
- [RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, 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.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.
- [I-D.ietf-6lo-btle]
Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets over BLUETOOTH(R) Low Energy", draft-ietf-6lo-btle-03 (work in progress), September 2014.
- [CASE] Case, J. and C. Partridge, "Case Diagrams: A First Step to Diagrammed Management Information Bases", Computer Communications Review 19(1), January 1989.

Authors' Addresses

Juergen Schoenwaelder
Jacobs University
Campus Ring 1
Bremen 28759
Germany

EMail: j.schoenwaelder@jacobs-university.de

Anuj Sehgal
Jacobs University
Campus Ring 1
Bremen 28759
Germany

EMail: s.anuj@jacobs-university.de

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

EMail: tina.tsou.zouting@huawei.com

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

EMail: cathyzhou@huawei.com

IPv6 over Networks of Resource-constrained Nodes (6lo) WG A. Brandt
Internet-Draft J. Buron
Intended status: Standards Track Sigma Designs
Expires: May 3, 2015 October 30, 2014

Transmission of IPv6 packets over ITU-T G.9959 Networks
draft-ietf-6lo-lowpanz-08

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2015.

Copyright Notice

Copyright (c) 2014 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. Terms used	3
2. G.9959 parameters to use for IPv6 transport	5
2.1. Addressing mode	5
2.2. IPv6 Multicast support	6
2.3. G.9959 MAC PDU size and IPv6 MTU	6
2.4. Transmission status indications	7
2.5. Transmission security	7
3. 6LoWPAN Adaptation Layer and Frame Format	7
3.1. Dispatch Header	8
4. 6LoWPAN addressing	9
4.1. Stateless Address Autoconfiguration of routable IPv6 addresses	9
4.2. IPv6 Link Local Address	9
4.3. Unicast Address Mapping	10
4.4. On the use of Neighbor Discovery technologies	10
4.4.1. Prefix and CID management (Route-over)	11
4.4.2. Prefix and CID management (Mesh-under)	11
5. Header Compression	12
6. IANA Considerations	13
7. Security Considerations	13
8. Privacy Considerations	13
9. Acknowledgements	14
10. References	14
10.1. Normative References	14
10.2. Informative References	15
Appendix A. G.9959 6LoWPAN datagram example	16
Appendix B. Change Log	20
B.1. Changes since -00	20
B.2. Changes since -01	20
B.3. Changes since -02	21
B.4. Changes since -03	21
B.5. Changes since -04	22
B.6. Changes since -05	22
B.7. Changes since -06	22
B.8. Changes since -07	22
Authors' Addresses	23

1. Introduction

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

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

[RFC6775] updates [RFC4944] by specifying 6LoWPAN optimizations for IPv6 Neighbor Discovery (ND) (originally defined by [RFC4861]). This document limits the use of [RFC6775] to prefix and Context ID assignment. An IID may be constructed from a G.9959 link-layer address, leading to a "link-layer-derived IPv6 address". If using that method, Duplicate Address Detection (DAD) is not needed. Alternatively, IPv6 addresses may be assigned centrally via DHCP, leading to a "non-link-layer-derived IPv6 address". Address registration is only needed in certain cases.

In addition to IPv6 application communication, the frame format defined in this document may be used by IPv6 routing protocols such as RPL [RFC6550] or P2P-RPL [RFC6997] to implement IPv6 routing over G.9959 networks.

The encapsulation frame defined by this specification may optionally be transported via mesh routing below the 6LoWPAN layer. Mesh-under and route-over routing protocol specifications are out of scope of this document.

1.1. Terms used

6LoWPAN: IPv6-based Low-power Personal Area Network

ABR: Authoritative 6LBR ([RFC6775])

Ack: Acknowledgement

AES: Advanced Encryption Scheme

CID: Context Identifier ([RFC6775])

DAD: Duplicate Address Detection ([RFC6775])

DHCPv6: Dynamic Host Configuration Protocol for IPv6 ([RFC3315])

EUI-64: Extended Unique Identifier ([EUI64])

G.9959: Short range, narrow-band digital radiocommunication transceiver ([G.9959])

GHC: Generic Header Compression ([RFC_TBD_GHC])

HomeID: G.9959 Link-Layer Network Identifier

IID: Interface IDentifier

Link-layer-derived address: IPv6 Address constructed on basis of link layer address information

MAC: Media Access Control

Mesh-under: Forwarding via mesh routing below the 6LoWPAN layer

MTU: Maximum Transmission Unit

ND: Neighbor discovery ([RFC4861], [RFC6775])

NodeID: G.9959 Link-Layer Node Identifier

Non-link-layer-derived address: IPv6 Address assigned by a managed process, e.g. DHCPv6.

NVM: Non-volatile Memory

P2P-RPL: Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks ([RFC6997])

PAN: Personal Area Network

PDU: Protocol Data Unit

PHY: Physical Layer

RA: Router Advertisement ([RFC4861], [RFC6775])

Route-over: Forwarding via IP routing above the 6LoWPAN layer

RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks ([RFC6550])

SAR: G.9959 Segmentation And Reassembly

ULA: Unique Local Address [RFC4193]

2. G.9959 parameters to use for IPv6 transport

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

2.1. Addressing mode

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

An IPv6 host MUST construct its link-local IPv6 address from the link-layer-derived IID in order to facilitate IP header compression as described in [RFC6282].

A node interface MAY support the M flag of the RA message for the construction of routable IPv6 addresses. A cost optimized node implementation may save memory by skipping support for the M flag. The M flag MUST be interpreted as defined in Figure 1.

M Flag support	M flag value	Required node behavior
No	(ignore)	Node MUST use link-layer-derived addressing
Yes	0	Node MUST use link-layer-derived addressing
	1	Node MUST use DHCPv6 based addressing and Node MUST comply fully with [RFC6775]

Figure 1: RA M flag support and interpretation

A node that uses DHCPv6 based addressing MUST comply fully with the text of [RFC6775].

If DHCPv6 based addressing is used, the DHCPv6 client must use a DUID of type DUID-UUID, as described in [RFC6355]. The UUID used in the DUID-UUID must be generated as specified in [RFC4122], section 4.5, starting at the second paragraph in that section (the 47-bit random number-based UUID). The DUID must be stored persistently by the node as specified in section 3 of [RFC6355].

A word of caution: since HomeIDs and NodeIDs are handed out by a network controller function during inclusion, identifier validity and

uniqueness is limited by the lifetime of the network membership. This can be cut short by a mishap occurring to the network controller. Having a single point of failure at the network controller suggests that high-reliability network deployments may benefit from a redundant network controller function.

This warning applies to link-layer-derived addressing as well as to non-link-layer-derived addressing deployments.

2.2. IPv6 Multicast support

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

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

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

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

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

2.3. G.9959 MAC PDU size and IPv6 MTU

IPv6 packets MUST be transmitted using G.9959 transmission profile R3 or higher.

[RFC2460] specifies that any link that cannot convey a 1280-octet packet in one piece, must provide link-specific fragmentation and reassembly at a layer below IPv6.

G.9959 provides Segmentation And Reassembly for payloads up to 1350 octets. IPv6 Header Compression [RFC6282] improves the chances that a short IPv6 packet can fit into a single G.9959 frame. Therefore, Section 3 specifies that [RFC6282] MUST be supported. With the mandatory link-layer security enabled, a G.9959 R3 MAC PDU may accommodate 6LoWPAN datagrams of up to 130 octets without triggering G.9959 Segmentation and Reassembly (SAR). Longer 6LoWPAN datagrams will lead to the transmission of multiple G.9959 PDUs.

2.4. Transmission status indications

The G.9959 MAC layer provides native acknowledgement and retransmission of MAC PDUs. The G.9959 SAR layer does the same for larger datagrams. A mesh routing layer may provide a similar feature for routed communication. An IPv6 routing stack communicating over G.9959 may utilize link-layer status indications such as delivery confirmation and Ack timeout from the MAC layer.

2.5. Transmission security

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

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

3. 6LoWPAN Adaptation Layer and Frame Format

The 6LoWPAN encapsulation formats defined in this chapter are carried as payload in the G.9959 MAC PDU. IPv6 header compression [RFC6282] MUST be supported by implementations of this specification. Further, implementations MAY support Generic Header Compression (GHC) [RFC_TBD_GHC]. A node implementing [RFC_TBD_GHC] MUST probe its peers for GHC support before applying GHC compression.

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

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

An example of a complete G.9959 6LoWPAN datagram can be found in Appendix A.

3.1. Dispatch Header

The dispatch header is shown below:

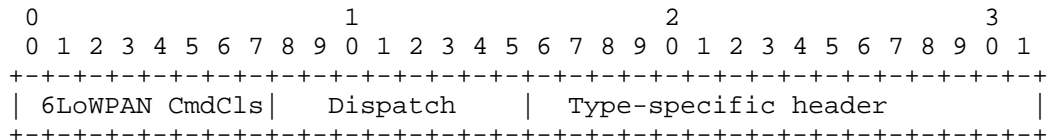


Figure 2: Dispatch Type and Header

6LoWPAN CmdCls: 6LoWPAN Command Class identifier. This field **MUST** carry the value 0x4F [G.9959]. The value is assigned by the ITU-T and specifies that the following bits are a 6LoWPAN encapsulated datagram. 6LoWPAN protocols **MUST** ignore the G.9959 frame if the 6LoWPAN Command Class identifier deviates from 0x4F.

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

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

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

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

Pattern	Header Type	Reference
01 1xxxxx	6LoWPAN_IPHC - Compressed IPv6 Addresses	[RFC6282]

All other Dispatch values are unassigned in this document.

Figure 3: Dispatch values

6LoWPAN_IPHC: IPv6 Header Compression. Refer to [RFC6282].

4. 6LoWPAN addressing

IPv6 addresses may be autoconfigured from IIDs which may again be constructed from link-layer address information to save memory in devices and to facilitate efficient IP header compression as per [RFC6282]. Link-layer-derived addresses have a static nature and may involuntarily expose private usage data on public networks. Refer to Section 8.

A NodeID is mapped into an IEEE EUI-64 identifier as follows:

IID = 0000:00ff:fe00:YYXX

Figure 4: Constructing a compressible IID

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

The method of constructing IIDs from the link-layer address obviously does not support addresses assigned or constructed by other means. A node MUST NOT compute the NodeID from the IID if the first 6 bytes of the IID do not comply with the format defined in Figure 4. In that case, the address resolution mechanisms of RFC 6775 apply.

4.1. Stateless Address Autoconfiguration of routable IPv6 addresses

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

4.2. IPv6 Link Local Address

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

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

The resulting link local address is formed as follows:

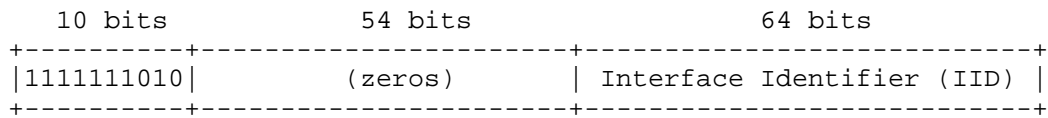


Figure 5: IPv6 Link Local Address

4.3. Unicast Address Mapping

The address resolution procedure for mapping IPv6 unicast addresses into G.9959 link-layer addresses follows the general description in Section 7.2 of [RFC4861]. The Source/Target Link-layer Address option MUST have the following form when the link layer is G.9959.

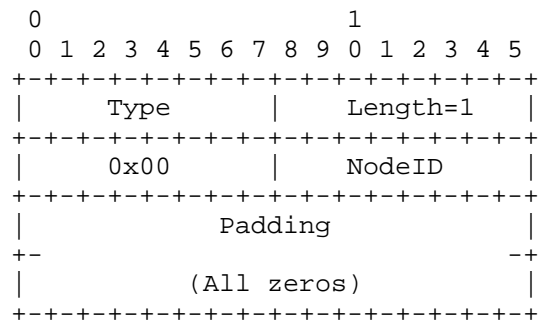


Figure 6: IPv6 Unicast Address Mapping

Option fields:

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

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

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

4.4. On the use of Neighbor Discovery technologies

[RFC4861] specifies how IPv6 nodes may resolve link layer addresses from IPv6 addresses via the use of link-local IPv6 multicast. [RFC6775] is an optimization of [RFC4861], specifically targeting

6LoWPAN networks. [RFC6775] defines how a 6LoWPAN node may register IPv6 addresses with an authoritative border router (ABR). Mesh-under networks MUST NOT use [RFC6775] address registration. However, [RFC6775] address registration MUST be used if the first 6 bytes of the IID do not comply with the format defined in Figure 3.

4.4.1. Prefix and CID management (Route-over)

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

4.4.2. Prefix and CID management (Mesh-under)

An implementation for mesh-under operation MUST use [RFC6775] mechanisms for managing IPv6 prefixes and corresponding header compression context information [RFC6282]. [RFC6775] Duplicate Address Detection (DAD) MUST NOT be used, since the link-layer inclusion process of G.9959 ensures that a NodeID is unique for a given HomeID.

With this exception and the specific redefinition of the RA Router Lifetime value 0xFFFF (refer to Section 4.4.2.3), the text of the following subsections is in compliance with [RFC6775].

4.4.2.1. Prefix assignment considerations

As stated by [RFC6775], an ABR is responsible for managing prefix(es). Global routable prefixes may change over time. It is RECOMMENDED that a ULA prefix is assigned to the 6LoWPAN subnet to facilitate stable site-local application associations based on IPv6 addresses. A node MAY support the M flag of the RA message. This influences the way IPv6 addresses are assigned. Refer to Section 2.1 for details.

4.4.2.2. Robust and efficient CID management

The 6LoWPAN Context Option (6CO) is used according to [RFC6775] in an RA to disseminate Context IDs (CID) to use for compressing prefixes. One or more prefixes and corresponding Context IDs MUST be assigned during initial node inclusion.

When updating context information, a CID may have its lifetime set to zero to obsolete it. The CID MUST NOT be reused immediately; rather the next vacant CID should be assigned. Header compression based on CIDs MUST NOT be used for RA messages carrying Context Information.

An expired CID and the associated prefix MUST NOT be reset but rather retained in receive-only mode if there is no other current need for the CID value. This will allow an ABR to detect if a sleeping node without clock uses an expired CID and in response, the ABR MUST return an RA with fresh Context Information to the originator.

4.4.2.3. Infinite prefix lifetime support for island-mode networks

Nodes MUST renew the prefix and CID according to the lifetime signaled by the ABR. [RFC6775] specifies that the maximum value of the RA Router Lifetime field MAY be up to 0xFFFF. This document further specifies that the value 0xFFFF MUST be interpreted as infinite lifetime. This value MUST NOT be used by ABRs. Its use is only intended for a sleeping network controller; for instance a battery powered remote control being master for a small island-mode network of light modules.

5. Header Compression

IPv6 header compression [RFC6282] MUST be implemented and [RFC_TBD_GHC] compression for higher layers MAY be implemented. This section will simply identify substitutions that should be made when interpreting the text of [RFC6282] and [RFC_TBD_GHC].

In general the following substitutions should be made:

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

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

```

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

```

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

64 bit 802.15.4 address details do not apply.

6. IANA Considerations

This document makes no request of IANA.

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

7. Security Considerations

The method of derivation of Interface Identifiers from 8-bit NodeIDs preserves uniqueness within the network. However, there is no protection from duplication through forgery. Neighbor Discovery in G.9959 links may be susceptible to threats as detailed in [RFC3756]. G.9959 networks may feature mesh routing. This implies additional threats due to ad hoc routing as per [KW03]. G.9959 provides capability for link-layer security. G.9959 nodes MUST use link-layer security with a shared key. Doing so will alleviate the majority of threats stated above. A sizeable portion of G.9959 devices is expected to always communicate within their PAN (i.e., within their subnet, in IPv6 terms). In response to cost and power consumption considerations, these devices will typically implement the minimum set of features necessary. Accordingly, security for such devices may rely on the mechanisms defined at the link layer by G.9959. G.9959 relies on the Advanced Encryption Standard (AES) for authentication and encryption of G.9959 frames and further employs challenge-response handshaking to prevent replay attacks.

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

8. Privacy Considerations

IP addresses may be used to track devices on the Internet, which in turn can be linked to individuals and their activities. Depending on the application and the actual use pattern, this may be undesirable. To impede tracking, globally unique and non-changing characteristics of IP addresses should be avoided, e.g. by frequently changing the global prefix and avoiding unique link-layer-derived IIDs in addresses.

Some link layers use a 48-bit or a 64-bit link layer address which uniquely identifies the node on a global scale regardless of global

prefix changes. The risk of exposing a G.9959 device from its link-layer-derived IID is limited because of the short 8-bit link layer address.

While intended for central address management, DHCPv6 address assignment also decouples the IPv6 address from the link layer address. Addresses may be made dynamic by the use of a short DHCP lease period and an assignment policy which makes the DHCP server hand out a fresh IP address every time. For enhanced privacy, the DHCP assigned addresses should be logged only for the duration of the lease provided the implementation also allows logging for longer durations as per the operational policies.

It should be noted that privacy and frequently changing address assignment comes at a cost. Non-link-layer-derived IIDs require the use of address registration and further, non-link-layer-derived IIDs cannot be compressed, which leads to longer datagrams and increased link layer segmentation. Finally, frequent prefix changes necessitate more Context Identifier updates, which not only leads to increased traffic but also may affect the battery lifetime of sleeping nodes.

9. Acknowledgements

Thanks to the authors of RFC 4944 and RFC 6282 and members of the IETF 6LoWPAN working group; this document borrows extensively from their work. Thanks to Erez Ben-Tovim, Erik Nordmark, Kerry Lynn, Michael Richardson, Tommas Jess Christensen for useful comments. Thanks to Carsten Bormann for extensive feedback which improved this document significantly. Thanks to Brian Haberman for pointing out unclear details.

10. References

10.1. Normative References

- [G.9959] "G.9959 (02/12) + G.9959 Amendment 1 (10/13): Short range, narrow-band digital radiocommunication transceivers", February 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, August 2011.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.
- [RFC_TBD_GHC] "draft-ietf-6lo-ghc: 6LoWPAN Generic Compression of Headers and Header-like Payloads", September 2014.

10.2. Informative References

- [EUI64] IEEE, "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", IEEE Std <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>, November 2012.
- [KW03] Elsevier's AdHoc Networks Journal, "Secure Routing in Sensor Networks: Attacks and Countermeasures", Special Issue on Sensor Network Applications and Protocols vol 1, issues 2-3", , September 2003.

- [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.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, August 2003.
- [RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.
- [RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.
- [RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.
- [RFC6997] Goyal, M., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks", RFC 6997, August 2013.

Appendix A. G.9959 6LoWPAN datagram example

This example outlines each individual bit of a sample IPv6 UDP packet arriving to a G.9959 node from a host in the Internet via a PAN border router.

In the G.9959 PAN, the complete frame has the following fields.

G.9959:

```
+-----+-----+-----+-----+-----+-----+...
|HomeID|SrcNodeID|FrmControl|Len|SeqNo|DestNodeID|
+-----+-----+-----+-----+-----+-----+...
```

6LoWPAN:

```
...+-----+-----+-----+-----+-----+-----+...
|6LoWPAN CmdCls|6LoWPAN_IPHC Hdr|Compressed IPv6 headers|
...+-----+-----+-----+-----+-----+-----+...
```

6LoWPAN, TCP/UDP, App payload:

```
...+-----+-----+-----+-----+-----+-----+
|Uncompressed IPv6 headers|TCP/UDP/ICMP|App payload|
...+-----+-----+-----+-----+-----+-----+
```

The frame comes from the source IPv6 address
2001:0db8:ac10:ef01::ff:fe00:1206. The source prefix
2001:0db8:ac10:ef01/64 is identified by the IPHC CID = 3.

The frame is delivered in direct range from the gateway which has
source NodeID = 1. The Interface Identifier (IID) ff:fe00:1206 is
recognised as a link-layer-derived address and is compressed to the
16 bit value 0x1206.

The frame is sent to the destination IPv6 address
2001:0db8:27ef:42ca::ff:fe00:0004. The destination prefix
2001:0db8:27ef:42ca/64 is identified by the IPHC CID = 2.

The Interface Identifier (IID) ff:fe00:0004 is recognised as a link-
layer-derived address.

Thanks to the link-layer-derived addressing rules, the sender knows
that this is to be sent to G.9959 NodeID = 4; targeting the IPv6
interface instance number 0 (the default).

To reach the 6LoWPAN stack of the G.9959 node, (skipping the G.9959
header fields) the first octet must be the 6LoWPAN Command Class
(0x4F).

```

0
0 1 2 3 4 5 6 7 8
+---+---+---+---+---+---+...
|       0x4F       |
+---+---+---+---+---+---+...

```

The Dispatch header bits '011' advertises a compressed IPv6 header.

```

0                               1
0 1 2 3 4 5 6 7 8 9 0
+---+---+---+---+---+---+---+...
|       0x4F       | 0 1 1
+---+---+---+---+---+---+---+...

```

The following bits encode the first IPv6 header fields:

TF = '11' : Traffic Class and Flow Label are elided.
 NH = '1' : Next Header is elided
 HLIM = '10' : Hop limit is 64

```

0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+...
|       0x4F       | 0 1 1 1 1 1 1 0 |
+---+---+---+---+---+---+---+---+...

```

CID = '1' : CI data follows the DAM field
 SAC = '1' : Src addr uses stateful, context-based compression
 SAM = '10' : Use src CID and 16 bits for link-layer-derived addr
 M = '0' : Dest addr is not a multicast addr
 DAC = '1' : Dest addr uses stateful, context-based compression
 DAM = '11' : Use dest CID and dest NodeID to link-layer-derived addr

```

0                               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
+---+---+---+---+---+---+---+---+---+---+---+---+---+...
|       0x4F       | 0 1 1 1 1 1 1 0 | 1 1 1 0 0 1 1 1 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+...

```

Address compression context identifiers:

SCI = 0x3

DCI = 0x2

```

      2          3
    4 5 6 7 8 9 0 1
...+--+--+--+--+--+...
  | 0x3 | 0x2 |
...+--+--+--+--+--+...
```

IPv6 header fields:

(skipping "version" field)

(skipping "Traffic Class")

(skipping "flow label")

(skipping "payload length")

IPv6 header address fields:

SrcIP = 0x1206 : Use SCI and 16 LS bits of link-layer-derived address

(skipping DestIP) - completely reconstructed from Dest NodeID and DCI

```

      2          3          4
    4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
...+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+...
  | 0x3 | 0x2 | 0x12 | 0x06 |
...+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+...
```

Next header encoding for the UDP header:

Dispatch = '11110': Next Header dispatch code for UDP header

C = '0' : 16 bit checksum carried inline

P = '00' : Both src port and dest Port are carried in-line.

```

      4    5
    8 9 0 1 2 3 4 5
...+--+--+--+--+--+...
  | 1 1 1 1 0 | 0 | 0 |
...+--+--+--+--+--+...
```

UDP header fields:

src Port = 0x1234

dest port = 0x5678

```

      5         6             7             8
      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 2 3 4 5 6 7
...+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+...
  |      0x12      |      0x34      |      0x56      |      0x78      |
...+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+...

```

(skipping "length")

checksum = (actual checksum value depends on
the actual UDP payload)

```

                        1
                        0
      8  9             0
      8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
...+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+...
  |      (UDP checksum)      |
...+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+...

```

Add your own UDP payload here...

Appendix B. Change Log

B.1. Changes since -00

- o Clarified that mesh-under routing may take place below the 6LoWPAN layer but that specific mesh-under routing protocols are not within the scope of this doc.
- o Clarified that RFC6282 IPv6 Header Compression MUST be supported.
- o Clarified the text of section 5.4 on the use of RFC6775 address registration in mesh-under networks.
- o Split 5.4.2 into multiple paragraphs.

B.2. Changes since -01

- o Added this Change Log
- o Editorial nits.

- o Made IPv6 Header Compression mandatory. Therefore, the Dispatch value "01 000001 - Uncompressed IPv6 Addresses" was removed from figure 2.
- o Changed SHOULD to MUST: An IPv6 host SHOULD construct its link-local IPv6 address and routable IPv6 addresses from the NodeID in order to facilitate IP header compression as described in [RFC6282].
- o Changed SHOULD NOT to MUST NOT: Mesh-under networks MUST NOT use [RFC6775] address registration.
- o Changed SHOULD NOT to MUST NOT: [RFC6775] Duplicate Address Detection (DAD) MUST NOT be used.
- o Changed SHOULD NOT to MUST NOT: The CID MUST NOT be reused immediately;
- o Changed SHOULD NOT to MUST NOT: An expired CID and the associated prefix MUST NOT be reset but rather retained in receive-only mode
- o Changed LBR -> ABR
- o Changed SHOULD to MUST: , the ABR MUST return an RA with fresh Context Information to the originator.
- o Changed SHOULD NOT to MUST NOT: This value MUST NOT be used by ABRs. Its use is only intended for a sleeping network controller.

B.3. Changes since -02

- o Editorial nits.
- o Moved text to the right section so that it does not prohibit DAD for Route-Over deployments.
- o Introduced RA M flag support so that nodes may be instructed to use DHCPv6 for centralized address assignment.
- o Added example appendix: Complete G.9959 6LoWPAN datagram composition with CID-based header compression.

B.4. Changes since -03

- o Corrected error in 6LoWPAN datagram example appendix: 64 hop limit in comment => also 64 hop limit in actual frame format.
- o Added section "Privacy Considerations"

B.5. Changes since -04

- o Text on RA M flag support was replaced with a table to improve clarity.
- o Added further details to text on achieving privacy addressing with DHCP.

B.6. Changes since -05

- o Term ABR now points to Authoritative 6LBR as defined by RFC6775.
- o Removed sentence "The G.9959 network controller function SHOULD be integrated in the ABR." as this was an implementation guideline with no relevance to network performance.
- o Clarifying that network controller function redundancy is relevant for network deployers; not user-level application designers.
- o Clarified that RFC2460 specifies that link layer must provide fragmentation if 1280 octet packets cannot be carried in one piece by the link layer.
- o Clarified that the 6LoWPAN CmdCls identifier value is assigned by the ITU-T.
- o Added reference to Privacy Considerations section from 6LoWPAN Addressing section.
- o Introducing optional GHC header compression.

B.7. Changes since -06

- o Added a note to section 5, that the mapping of 802.15.4 terms to similar G.9959 terms applies not only to RFC6282 but also to GHC.

B.8. Changes since -07

- o Added a note to the Privacy considerations section on avoiding DHCP logging.
- o Added requirements for forming a UUID if DHCPv6 address assignment is used.

Authors' Addresses

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

Email: anders_brandt@sigmadesigns.com

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

Email: jakob_buron@sigmadesigns.com

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

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

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

Abstract

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

Status of this Memo

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

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

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

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

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

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

1. Introduction

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

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

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

1.1. Requirements Language

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

1.2. Abbreviations Used

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

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

CRC: Cyclic Redundancy Check

MAC: Medium Access Control

MSDU: MAC Service Data Unit (MAC client data)

UART: Universal Asynchronous Transmitter/Receiver

1.3. MS/TP Overview

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

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

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

MS/TP frames have the following format*:

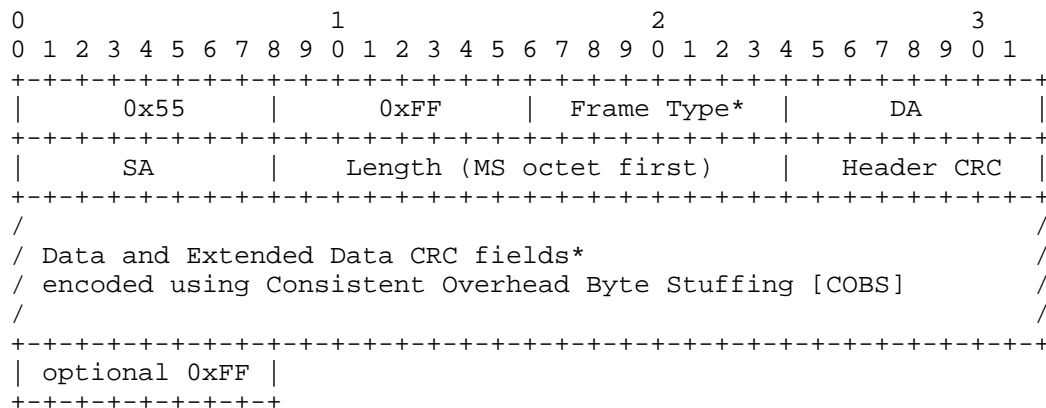


Figure 1: MS/TP Extended Frame Format

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

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

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

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

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

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

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

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

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

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

1.4. Goals and Non-goals

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

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

2. MS/TP Mode for IPv6

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

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

3. Addressing Modes

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

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

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

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

4. Maximum Transmission Unit (MTU)

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

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

5. LoBAC Adaptation Layer

The encapsulation formats defined in this section (subsequently referred to as the "LoBAC" encapsulation) comprise the payload (MSDU) of an MS/TP frame. The LoBAC payload (i.e., an IPv6 packet) follows an encapsulation header stack. LoBAC is a subset of the LOWPAN encapsulation defined in [RFC4944], therefore the use of "LOWPAN" in literals below is intentional. The primary differences between LoBAC and LOWPAN are: a) exclusion of the Fragmentation, Mesh, and Broadcast headers, and b) use of LOWPAN_IPHC [RFC6282] in place of LOWPAN_HC1 header compression (which is deprecated by [RFC6282]).

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

A LoBAC encapsulated IPv6 datagram:

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

A LoBAC encapsulated LOWPAN_IPHC compressed IPv6 datagram:

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

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

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

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

5.1. Dispatch Value and Header

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

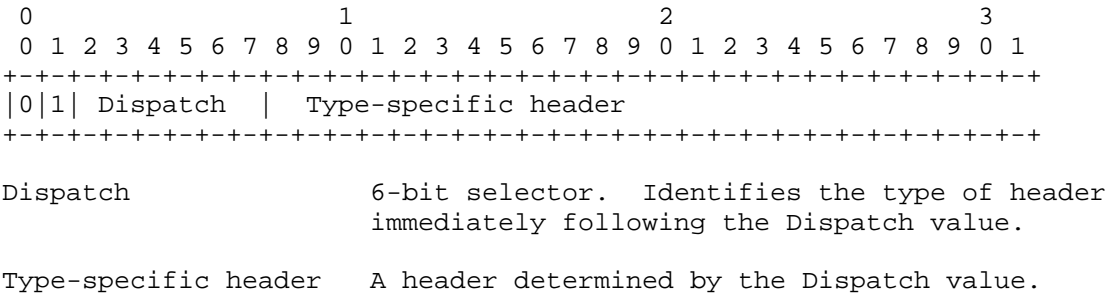


Figure 2: Dispatch Value and Header

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

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

Figure 3: Dispatch Value Bit Patterns

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

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

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

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

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

6. Stateless Address Autoconfiguration

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

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

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

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

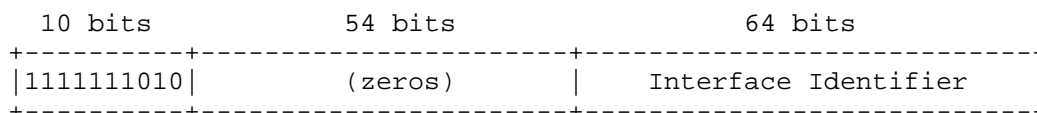
Note that this results in the universal/local bit set to "0" to indicate local scope.

An IPv6 address prefix used for stateless autoconfiguration [RFC4862]

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

7. IPv6 Link Local Address

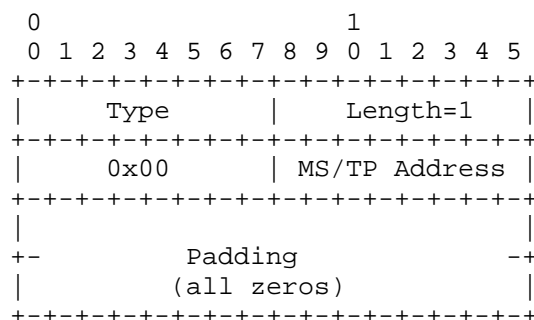
The IPv6 link-local address [RFC4291] for an MS/TP interface is formed by appending the Interface Identifier, as defined above, to the prefix FE80::/64.



8. Unicast Address Mapping

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

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



Option fields:

Type:

1: for Source Link-layer address.

2: for Target Link-layer address.

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

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

9. Multicast Address Mapping

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

```

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

```

10. Header Compression

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

In general the following substitutions should be made:

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

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

```

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

```

11. IANA Considerations

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

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

12. Security Considerations

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

13. Acknowledgments

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

14. References

14.1. Normative References

- [BACnet] American Society of Heating, Refrigerating, and Air-Conditioning Engineers, "BACnet, A Data Communication Protocol for Building Automation and Control Networks (ANSI Approved)", ANSI/ASHRAE 135-2010, April 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

14.2. Informative References

[Addendum_an]

ASHRAE, "BSR/ASHRAE Addendum an to ANSI/ASHRAE Standard 135-2010, BACnet - A Data Communication Protocol for Building Automation and Control Networks (Advisory Public Review Draft)", September 2011, <<https://osr.ashrae.org/default.aspx>>.

[COBS]

Cheshire, S. and M. Baker, "Consistent Overhead Byte Stuffing", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL.7, NO.2 , April 1999, <<http://www.stuartcheshire.org/papers/COBSforToN.pdf>>.

[CRC32K]

Koopman, P., "32-Bit Cyclic Redundancy Codes for Internet Applications", IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2002) , June 2002, <http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf>.

[EUI-64]

IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", March 1997, <<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>>.

[RFC2469]

Narten, T. and C. Burton, "A Caution On The Canonical Ordering Of Link-Layer Addresses", RFC 2469, December 1998.

[TIA-485-A]

Telecommunications Industry Association, "TIA-485-A, Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems (ANSI/TIA/EIA-485-A-98) (R2003)", March 2003.

Appendix A. Extended Data CRC [CRC32K]

This Appendix is informative and not part of the standard.

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

The BACnet MS/TP change proposal [Addendum_an] specifies the CRC32K (Koopman) polynomial. An example C implementation is shown below. The specified use of the function is that 'crcValue' is initialized to all ones before the function is first called and, upon running the function over all octets in the payload, the ones complement of 'crcValue' be sent in LSB-first order. Upon reception, the data field and modified 'crcValue' are passed again through the function. If these fields were properly received, the result of the function will be 'CRC32K_RESIDUE'.

```
#include <stdint.h>

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

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

    data = dataValue;
    crc = crcValue;

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

Appendix B. Consistent Overhead Byte Stuffing [COBS]

This Appendix is informative and not part of the standard.

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

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

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

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

```
#include <stdint.h>

#define MSTP_PREAMBLE_55 (0x55)

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

    uint32_t crc32K = CRC32K_INITIAL_VALUE;

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



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

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

```

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

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

        read_index++;

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

```

Authors' Addresses

Kerry Lynn (editor)
Consultant

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

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

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

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

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

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

Email: stuart.donaldson@honeywell.com

6Lo Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 8, 2014

P. Mariager, Ed.
J. Petersen
RTX A/S
Z. Shelby
Sensinode
M. Van de Logt
Gigaset Communications GmbH
D. Barthel
Orange Labs
May 7, 2014

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

Abstract

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

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on November 8, 2014.

Copyright Notice

Copyright (c) 2014 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. Requirements Notation	3
1.2. Terms Used	3
2. DECT Ultra Low Energy	4
2.1. The DECT ULE Protocol Stack	4
2.2. Link layer roles and topology	5
2.3. Addressing Model	6
2.4. MTU Considerations	7
2.5. Additional Considerations	7
3. Specification of IPv6 over DECT ULE	7
3.1. Protocol stack	8
3.2. Link model	8
3.3. Internet connectivity scenarios	12
4. IANA Considerations	13
5. Security Considerations	13
6. ETSI Considerations	13
7. Acknowledgements	13
8. Normative References	13
Authors' Addresses	15

1. Introduction

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

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

1.1. Requirements Notation

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

1.2. Terms Used

PP: DECT Portable Part, typically the sensor node

FP: DECT Fixed Part, the gateway

LLME: Lower Layer Management Entity

NWK: Network

PVC: Permanent Virtual Circuit

FAR: Fragmentation and Reassembly

2. DECT Ultra Low Energy

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

2.1. The DECT ULE Protocol Stack

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

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

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

The current DECT ULE MAC layer standard supports low bandwidth data broadcast. However the usage of this broadcast service has not yet been standardized for higher layers. This document is not considering usage of this DECT ULE MAC broadcast service in current version.

In general, communication sessions can be initiated from both FP and PP side. Depending of power down modes employed in the PP, latency may occur when initiating sessions from FP side. MAC layer communication can either take place using connection oriented packet

transfer with low overhead for short sessions or take place using connection oriented bearers including media reservation. The MAC layer autonomously selects the radio spectrum positions that are available within the band and can rearrange these to avoid interference. The MAC layer has built-in retransmission procedures in order to improve transmission reliability.

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

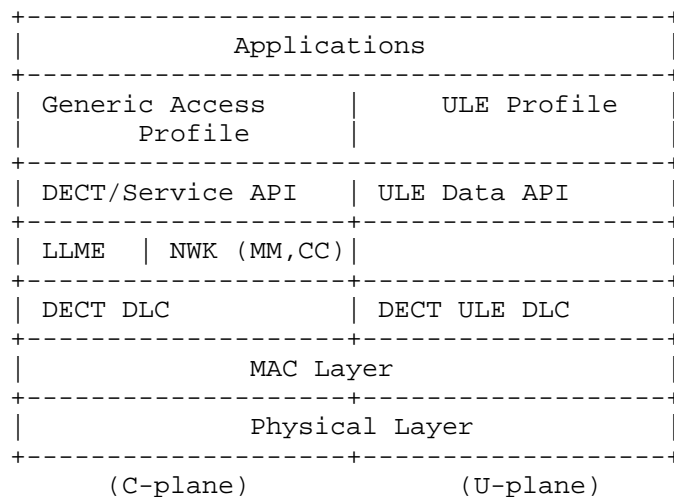


Figure 1: DECT ULE Protocol Stack

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

2.2. Link layer roles and topology

A FP is assumed to be less constrained than a PP. Hence, in the primary scenario FP and PP will act as 6LoWPAN Border Router (6LBR)

and a 6LoWPAN Node (6LN), respectively. This document does only address this primary scenario.

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

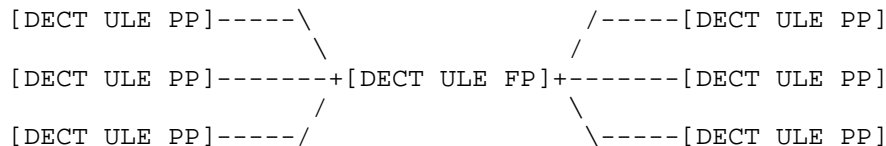


Figure 2: DECT ULE star topology

DECT ULE repeaters are not considered in this proposal.

2.3. Addressing Model

Each DECT PP is assigned an <IPEI> (International Portable Equipment Identity) during manufacturing. This identity has the size of 40 bits and is DECT globally unique for the PP and can be used to constitute the MAC address. However, it can not be used to derive a globally unique IID.

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

Each DECT FP is assigned a <RFPI> (Radio Fixed Part Identity) during manufacturing. This identity has the size of 40 bits and is globally unique for a FP and can be used to constitute the MAC address. However, it can not be used to derive a globally unique IID.

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

2.4. MTU Considerations

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

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

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

2.5. Additional Considerations

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

3. Specification of IPv6 over DECT ULE

DECT ULE technology sets strict requirements for low power consumption and thus limits the allowed protocol overhead. 6LoWPAN standards [RFC4944], [RFC6775], and [RFC6282] provide useful functionality for reducing overhead which can be applied to DECT ULE. This functionality comprises link-local IPv6 addresses and stateless IPv6 address autoconfiguration, Neighbor Discovery and header compression.

The ULE 6LoWPAN adaptation layer can run directly on this U-plane DLC layer. Figure 3 illustrates IPv6 over DECT ULE stack.

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

3.1. Protocol stack

In order to enable transmission of IPv6 packets over DECT ULE, a Permanent Virtual Circuit (PVC) has to be opened between FP and PP. This MUST be done by setting up a service call from PP to FP. The PP SHALL specify the <<IWU-ATTRIBUTES>> in a service-change (other) message before sending a service-change (resume) message as defined in [TS102.939-1]. The <<IWU-ATTRIBUTES>> SHALL define the ULE Application Protocol Identifier to 0x06 (reserved for 6LoWPAN and has to be standardized by ETSI) and the MTU size to 1280 octets or larger. The FP MUST send a service-change-accept (resume) containing a valid paging descriptor. The PP MUST be pageable.

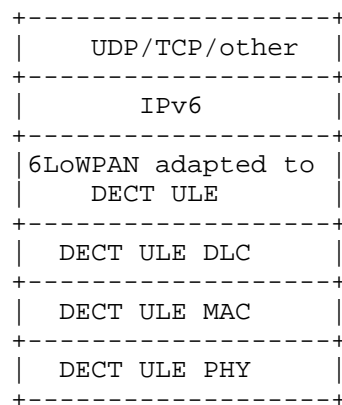


Figure 3: IPv6 over DECT ULE Stack

3.2. Link model

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

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

Due to DECT ULE star topology, each branch of the star is considered to be an individual link and thus the PPs cannot directly hear one another and also cannot talk to one another with link-local

addresses. After the FP and PPs have connected at the DECT ULE level, the link can be considered up and IPv6 address configuration and transmission can begin. The FP ensures address collisions do not occur.

3.2.1. Stateless address autoconfiguration

A DECT ULE 6LN performs stateless address autoconfiguration as per [RFC4862]. A 64-bit Interface identifier (IID) for a DECT ULE interface MAY be formed by utilizing a MAC-48 device address as defined in [RFC2464] "IPv6 over Ethernet" specification. Alternatively, the DECT device addresses IPEI, RFPI or TPUI, MAY be used instead to derive the IID. In the case of randomly generated IID or use of IID derived from DECT devices addresses, the "Universal/Local" bit MUST be set to 0. Only if a global unique MAC-48 is used the "Universal/Local" bit can be set to 1.

As defined in [RFC4291], the IPv6 link-local address for a DECT ULE node is formed by appending the IID, to the prefix FE80::/64, as shown in Figure 4.

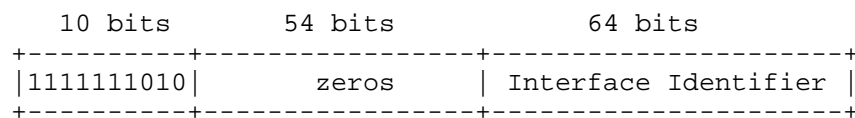


Figure 4: IPv6 link-local address in DECT ULE

The means for a 6LBR to obtain an IPv6 prefix for numbering the DECT ULE network is out of scope of this document, but can be, for example, accomplished via DHCPv6 Prefix Delegation [RFC3633] or by using Unique Local IPv6 Unicast Addresses (ULA) [RFC4193]. Due to the link model of the DECT ULE the 6LBR MUST set the "on-link" flag (L) to zero in the Prefix Information Option [RFC4861]. This will cause 6LNs to always send packets to the 6LBR, including the case when the destination is another 6LN using the same prefix.

3.2.2. Neighbor discovery

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

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

1. A DECT ULE 6LN MUST register its address with the 6LBR by sending a Neighbor Solicitation (NS) message with the ARO option and process the Neighbor Advertisement (NA) accordingly. The NS with the ARO option SHOULD be sent irrespective of whether the IID is derived from a unique MAC-48 bit device address, from the DECT ULE device addresses or the IID is a random value that is generated as per the privacy extensions for stateless address autoconfiguration [RFC4941]. Although [RFC4941] permits the use of deprecated addresses for old connections, in this specification we mandate that one interface MUST NOT use more than one IID at any one time.

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

3.2.3. Unicast and Multicast address mapping

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

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

3.2.4. Header Compression

Header compression as defined in [RFC6282], which specifies the compression format for IPv6 datagrams on top of IEEE 802.15.4, is REQUIRED in this document as the basis for IPv6 header compression on top of DECT ULE. All headers MUST be compressed according to [RFC6282] encoding formats. The DECT ULE's star topology structure can be exploited in order to provide a mechanism for IID compression. The following text describes the principles of IPv6 address compression on top of DECT ULE.

In a link-local communication, both the IPv6 source and destination addresses MUST be elided [RFC6282], since the node knows that the packet is destined for it even if the packet does not have destination IPv6 address. A node SHALL learn the IID of the other endpoint of each DECT ULE connection it participates in. By exploiting this information, a node that receives a PDU containing an IPv6 packet can infer the corresponding IPv6 source address. A node MUST maintain a Neighbor Cache, in which the entries include both the IID of the neighbor and the Device Address that identifies the neighbor. For the type of communication considered in this paragraph, the following settings MUST be used in the IPv6 compressed header: CID=0, SAC=0, SAM=11, DAC=0, DAM=11.

When a 6LN transmits an IPv6 packet to a remote destination using global Unicast IPv6 addresses, if a context is defined for the prefix of the 6LN's global IPv6 address, the 6LN MUST indicate this context in the corresponding source fields of the compressed IPv6 header as per Section 3.1 of [RFC6282], and MUST elide the IPv6 source address. For this, the 6LN MUST use the following settings in the IPv6 compressed header: CID=1, SAC=1, SAM=11. In this case, the 6LBR can infer the elided IPv6 source address since 1) the 6LBR has previously assigned the prefix to the 6LNs; and 2) the 6LBR maintains a Neighbor Cache that relates the Device Address and the IID of the corresponding PP. If a context is defined for the IPv6 destination address, the 6LN MUST also indicate this context in the corresponding destination fields of the compressed IPv6 header, and MUST elide the prefix of the destination IPv6 address. For this, the 6LN MUST set the DAM field of the compressed IPv6 header as DAM=01 (if the context covers a 64-bit prefix) or as DAM=11 (if the context covers a full, 128-bit address). CID and DAC MUST be set to CID=1 and DAC=1. Note that when a context is defined for the IPv6 destination address, the 6LBR can infer the elided destination prefix by using the context.

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

3.3. Internet connectivity scenarios

In a typical scenario, the DECT ULE network is connected to the Internet as shown in the Figure 5.

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

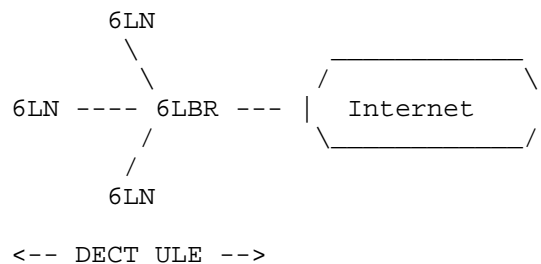


Figure 5: DECT ULE network connected to the Internet

In some scenarios, the DECT ULE network may transiently or permanently be an isolated network as shown in the Figure 6.

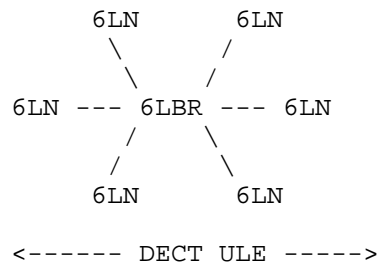


Figure 6: Isolated DECT ULE network

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

4. IANA Considerations

There are no IANA considerations related to this document.

5. Security Considerations

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

DECT ULE communications are secured at the link-layer (DLC) by encryption and per-message authentication through CCM mode (Counter with CBC-MAC) similar to [RFC3610]. The underlying algorithm for providing encryption and authentication is AES128.

The DECT ULE pairing procedure generates a master authentication key (UAK) and during location registration procedure or when the permanent virtual circuit are established, the session security keys are generated. Session security keys may be renewed regularly. The generated security keys (UAK and session security keys) are individual for each FP-PP binding, hence all PP in a system have different security keys. DECT ULE PPs do not use any shared encryption key.

6. ETSI Considerations

ETSI is standardizing a list of known application layer protocols that can use the DECT ULE permanent virtual circuit packet data service. Each protocol is identified by a unique known identifier, which is exchanged in the service-change procedure as defined in [TS102.939-1]. The IPv6/6LoWPAN as described in this document is considered as an application layer protocol on top of DECT ULE. In order to provide interoperability between 6LoWPAN / DECT ULE devices a common protocol identifier for 6LoWPAN has to be standardized by ETSI.

It is proposed to use ETSI DECT ULE Application Protocol Identifier equal 0x06 for 6LoWPAN.

7. Acknowledgements

8. Normative References

[EN300.175-part1-7]

ETSI, "Digital Enhanced Cordless Telecommunications (DECT); Common Interface (CI);", August 2013.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.
- [RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.
- [TS102.939-1] ETSI, "Digital Enhanced Cordless Telecommunications (DECT); Ultra Low Energy (ULE); Machine to Machine Communications; Part 1: Home Automation Network (phase 1)", April 2013.

Authors' Addresses

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

Email: pm@rtx.dk

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

Email: jtp@rtx.dk

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

Email: zach.shelby@sensinode.com

Marco van de Logt
Gigaset Communications GmbH
Frankenstrasse 2
D-46395 Bocholt
Germany

Email: marco.van-de-logt@gigaset.com

Dominique Barthel
Orange Labs

Email: dominique.barthel@orange.com

6lo
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2015

G. Rizzo, Ed.
AJ. Jara, Ed.
A. Olivieri
Y. Bocchi
HES-SO
MR. Palattella
SnT/Univ. of Luxembourg
L. Ladid
SnT/Univ. of Luxembourg/IPv6 Forum
S. Ziegler
C. Crettaz
Mandat International
March 28, 2015

IPv6 mapping to non-IP protocols
draft-rizzo-6lo-6legacy-03

Abstract

IPv6 is an important enabler of the Internet of Things, since it provides an addressing space large enough to encompass a vast and ubiquitous set of sensors and devices, allowing them to interconnect and interact seamlessly. To date, an important fraction of those devices is based on networking technologies other than IP. An important problem to solve in order to include them into an IPv6-based Internet of Things, is to define a mechanism for assigning an IPv6 address to each of them, in a way which avoids conflicts and protocol aliasing.

The only existing proposal for such a mapping leaves many problems unsolved and it is nowadays inadequate to cope with the new scenarios which the Internet of Things presents. This document defines a mechanism, 6TONon-IP, for assigning automatically an IPv6 address to devices which do not support IPv6 or IPv4, in a way which minimizes the chances of address conflicts, and of frequent configuration changes due to instability of connection among devices. Such a mapping mechanism enables stateless autoconfiguration for legacy technology devices, allowing them to interconnect through the Internet and to fully integrate into a world wide scale, IPv6-based IoT.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 29, 2015.

Copyright Notice

Copyright (c) 2015 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. Terminology	3
2. Introduction	3
2.1. Examples	4
2.1.1. Example 1 - Building automation systems and IoT . . .	4
2.1.2. Example 2 - KNX and demand-side management	5
3. Reference System	6
4. Issues addressed through the 6TONon-IP mapping mechanism . .	6
5. 6TONon-IP Mapping Method	8
6. Examples	9
6.1. Example 1 - EIB/KNX	9
6.2. Example 2 - RFID	10
7. IANA Considerations	10
8. Security considerations	10
9. Acknowledgements	11
10. Normative References	11
Authors' Addresses	11

1. 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].

2. Introduction

The Future Internet and the IPv6 protocol enable a new generation of techniques for accessing the network, which extend the Internet seamlessly to personal devices, sensors, home appliances, enabling the so called 'Internet of Things' (IoT). One of the key issues which presently hampers the development of IoT and limits its potential is the lack of an efficient common framework for the integration among the vast and diverse set of protocols and technologies which compose it. Current sensors and their application environments employ a large set of technologies which lack efficient interoperability. Some associations of manufacturers have been formed to build a common technological framework in specific application domains, e.g. KNX for building automation (<http://www.knx.org/>), ZigBee (ZigBee Alliance) (<http://www.zigbee.org/>), and protocols such as X10 and CAN. Such frameworks are based on very different architectures, and the protocols which compose them are generally not interoperable. Finally, most of these technologies were designed in a context of small and local networks, with limited capabilities, and they were not conceived for integration within the Internet. One of the ideas at the basis of the IoT is the constitution of a common set of protocols which enables the interaction between devices through the Internet. By enabling interaction through the Internet, new services could be conceived and implemented, increasing the value produced by the IoT infrastructure. The adoption of a common framework may make more economically convenient its deployment, and foster the development of new smart environments (buildings, cities, etc), ultimately making possible the full realization of the potential of the IoT. As deployment of new sensors is typically expensive, it is unthinkable of putting to disuse an installed set of sensors, once a new set of devices (typically, IPv6 enabled) is deployed. This is not an uncommon case, as the set of deployed legacy devices (sensors, actuators) is to date very large. Rather, mechanisms are needed to integrate legacy devices into a common IoT platform, in order to include them in all the present and future services (e.g. devices and services directory, localization services, etc) which will be implemented on the IoT. For these reasons, many designers of the Internet of Things are focusing on building such common access and communications framework. All the proposals (e.g. CoAP, RESTful Web services) presently under discussion are based on IPv6. This has important implications on the addressing of the devices. Indeed a

common addressing at the device level is mandatory, in order to implement true Machine to Machine (M2M) communications without Portal Servers, which would make the whole system difficult to integrate and scale. The present document focuses on the network layer aspects of such IPv6 based integration. At the network layer, a mechanism which assigns an IPv6 address to each device is needed, to solve the addressing problem. In this document, we propose a new mechanism for the users and devices to map the different addressing spaces to a common IPv6 one. Our proposed mechanism solves several issues posed by some of the mappings adopted so far. Such mapping makes it possible for every device from each technology to operate through a common framework based on IPv6 and protocols over IPv6 such as RESTful WebServices and Constrained Application Protocol (CoAP). For each technology, the proposed mechanism maps technology-specific features to a set of fields defined within the IPv6 address. This allows the location and identification of the devices in a multi-protocol card, or in any gateway or Portal Server.

2.1. Examples

In this subsection, we present two examples which help understanding the importance of adopting a common IPv6 based framework for interaction between things, and the need for legacy devices to be individually addressable through IPv6.

2.1.1. Example 1 - Building automation systems and IoT

The IoT is composed by a very large set of devices, which is poised to grow exponentially in the near future. For this reason, a directory service is needed, which offers the possibility to individuate a specific device or set of devices, with given capabilities or within a given geographical region. Let us assume such directory lists devices with their IPv6 addresses, and their function (say a temperature sensor, or a mobile phone, etc). For instance, let us consider the case of someone willing to build a map of temperatures in a given geographical region. Such directory service would allow retrieving the list of available devices within that region, each with its own IPv6 address. Assume some of those devices are legacy, non IP based temperature sensors and part of a given building automation system. Assume also that such system manages several of those temperature sensors. Even if such system would be reachable via IP, without having those sensors individually listed in the directory and appearing as "autonomous" things, which can be polled directly, one should resort to techniques for retrieving the temperature reading of those sensors which are specific of that building automation technology. This would make more complex the implementation of such a temperature map.

Instead, by having the building automation system expose each sensor as an IPv6 enabled device, the whole set of temperature sensors would be accessible in a homogeneous way, greatly simplifying the task.

2.1.2. Example 2 - KNX and demand-side management

KNX is a standardized (EN 50090, ISO/IEC 14543), OSI-based network communications protocol for intelligent buildings. Among the devices typically managed through KNX, we find:

- o Lighting control systems;
- o Heating/ventilation and air conditioning devices;
- o Shutter/blind and shading control systems; and
- o Energy management and electricity/gas/water metering devices.

KNX devices do not support IP. Therefore, in order to connect a KNX home network to the Internet, a gateway (KNXnet/IP router) is necessary. Other technologies for home automation are available nowadays, in which each smart device (air conditioners, washing machines, etc) supports IPv6. Let us consider a scenario in which an utility company offers an agreement to a fraction of its clients. In exchange for a cut on the energy bill, the utility company gains direct control over some appliances at the premises of the client. In this way, by powering off some of those devices in periods when the production cost of power are very high, the utility company realizes potentially high savings.

In order to implement this, the utility company sends commands to a set of devices under its direct control. For recently installed devices, the utility can assume that they support IPv6, and some application layer protocols such as CoAP. Therefore a command to switch off a device would use the IPv6 address to identify the device, and the application layer protocol to send the actual command. But for KNX devices, the command should have another format: the IPv6 address should be the one of the router bridging the IPv6 and the KNX networks, and upper layers protocols should take care of identifying the specific device inside the KNX home network to whom the command should be sent. Having to format a specific query for each specific home automation protocol adds a level of complexity which translates into higher costs of implementation and maintenance of such a service.

3. Reference System

In this section we describe a reference system where the IPv6 mapping is used. Such a system includes:

1. A set of networks running non-IPv6-compatible technologies, each with one or more hosts connected. Such networks generally use different OSI layer 3 protocols, or they may adopt a technology which does not have any layer 3 protocol.
2. A proxy, which hosts the IPv6 mapping functionality. Such device is typically connected to each of the legacy protocols networks, and it accesses the Internet via the IPv6 protocol. Such IPv6 addressing proxy performs all the necessary conversions and adaptations between IPv6 and the (local) networking protocol of the legacy technologies, in a way which depends on the specific legacy technology considered. This proxy makes use of the IPv6 mapping mechanism in order to transform the native addressing to IPv6 Host ID and vice versa in a way that depends on the legacy technology.

Though in what follows we will describe the proposed mapping with reference to such a system, the main ideas behind it are more general, and they apply to settings others than the one of reference presented here.

4. Issues addressed through the 6TONon-IP mapping mechanism

In this section we highlight the main open issues regarding assignment of IPv6 addresses to devices which do not support IPv6 or IPv4, and we describe a set of desirable properties for a mechanism for automatic assignment of IPv6 addresses to such devices, which we name henceforth 6TONon-IP. In Appendix A of RFC 4291, a method is described for creating modified EUI-64 format Interface Identifiers out of links or nodes with IEEE EUI-64 Identifiers, or with IEEE 802 48-bit MACs. Moreover, for technologies having other link layer interface identifier, some possible mapping methods are sketched, leaving for each legacy protocol the possibility to define its own mapping method.

In the present document, we propose a mapping mechanism which enables stateless address autoconfiguration for legacy technologies, and which exploits some protocol specific identifier such as link layer interface identifiers, and the like. The proposed mapping mechanism addresses the following issues:

1. Protocol identification: For the legacy protocols to which the mapping described in RFC 4291 does not apply, a mechanism is

needed to map an IPv6 address to the right legacy protocol. This feature is necessary in case of devices which operate as proxy for more than one legacy technology at the same time.

2. Inter protocol aliasing: Without a mechanism for identifying the legacy protocol from the host part of the IPv6 address, address conflicts are possible among devices belonging to different legacy protocols. For instance, this may happen when the link layer interface identifier is the same for two devices belonging to different technologies. As several legacy technologies are characterized by a small addressing space, address conflicts are not so unlikely.
3. Conflicts between IPv6 mapped legacy technology addresses and addresses derived from (modified or not) EUI-64 format interface identifiers.
4. Intra-protocol aliasing: As several legacy technologies are characterized by a small addressing space, it is not unlikely to have two legacy devices, mapped to IPv6 addresses with the same network ID (for instance, in the case in which they belong to two separate networks of the same technology, both connected to a same proxy), and with a same interface identifier, and mapping therefore to a same IPv6 address.

Moreover, the following is a list of desirable properties for a 6TONon-IP mapping:

1. Consistency: A host should get the same IPv6 address every time it connects to a same legacy network, assuming that the configuration of all the other devices in that network remains unchanged. This allows avoiding to advertise a new address every time the host reconnects. This feature might be particularly important for devices which are not always "on", or which are not permanently connected.
2. Local Uniqueness: For devices which have an IPv6 address with a same network part, the host part should be unique for each host. This property allows avoiding address conflicts.
3. Uniqueness within the whole Internet: Coherently with the IoT vision, the host part of an IPv6 address associated to a host should be unique within the whole Internet.

Depending on the specific legacy protocol, there might be protocol specific limitations to the satisfaction of these properties. In particular, for those protocols which do not have an interface identifier which is unique, properties 1) and 2) cannot be fully

satisfied. Indeed, no mapping can solve address conflicts which take place inside a legacy protocol network. When legacy protocols have a interface identifier which is unique, this can be used to produce a unique host part of an IPv6 address, and its uniqueness would guarantee the satisfaction of properties 1), 2) and 3).

5. 6TONon-IP Mapping Method

In this section we describe the proposed strategy for forming IPv6 addresses from legacy protocol information, and the address format that derives from it. We assume that (one or more) 64 bits Network ID prefixes are given to the mapping function, which therefore computes the 64 bits of the Host ID part of the address (IPv6 interface identifier), in order to form a full IPv6 address.

The input of the proposed mapping function consists in the interface identifier of the legacy protocol.

In the proposed mapping method, the resulting Host ID part (IPv6 interface identifier) is composed by six fields, as shown in Figure 1:

- o A Technology ID field (11 bits), containing a code which identifies the specific legacy protocol. This field is split into two parts, one of 6 bits, and another of 5 bits.
- o U/L bit (1 bit), in order to keep compatibility with the mapping EUI-64 [RFC4291]. The U/L bit is the seventh bit of the first byte and is used to determine whether the address is universally or locally administered. This bit is set to "0", in order to indicate local scope, analogously to what proposed in [RFC4291]. This choice prevents address conflicts with IPv6 interface identifier generated from IEEE EUI-64 identifiers or IEEE 48-bit MAC identifiers.
- o A Reserved field (4 bits). This field can be used in the future for the identification of different interfaces for a same technology (in the same subnetwork).
- o Technology Mapping field (32 bits), which maps the interface identifier of the legacy protocol. For those protocols for which the IID is not larger than 32 bits, this field contains the 32 bits of the IID. For IID which are larger than 32 bits, a hashing function is used instead of direct mapping. In particular, some hashing algorithms such as CRC-32 are suggested. Hashing satisfies the requirements of consistency and uniqueness within a subnet with a very high probability, which depends on the hashing

algorithm used. This field is split into two parts, one of 8 bits, and another of 24 bits.

- o The fourth and fifth bytes are both set to "0x00", in order not to conflict with EUI-64 interface identifiers.

The resulting format of the Host ID part of the IPv6 address obtained from the mapping is indicated in Figure 1.

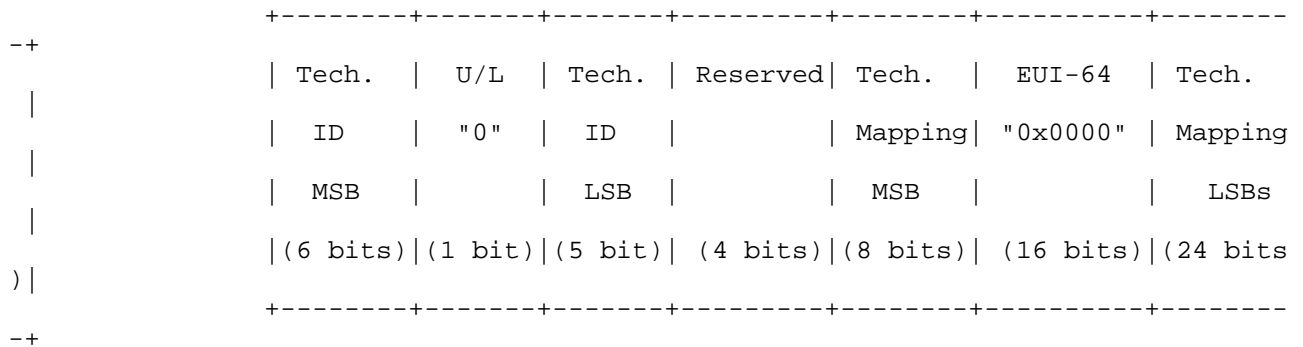


Figure 1: general format of the host ID part for legacy protocols

6. Examples

In this section we illustrate the proposed mapping method by applying it on some examples.

6.1. Example 1 - EIB/KNX

We assume the legacy protocol is EIB/KNX. This device has two kind of addresses: On the one hand, a logical address for management of group operations, and on the other hand, an individual address for identification of the device in the topology.

The mapping will be focused for the individual address. This includes an Area ID (4 bits), Line ID (8 bits), and Device ID (8 Bits). An example, is the value 0x1/0x01/0x01 for a sensor connected in the Area ID 0x1, Line ID 0x01, and Device ID 0x01.

We apply a hash (CRC-32) to the sequence 0x10101. The result is 0xDEA258A5.

Let us assume that EIB/Konnex Technology ID is "0". Thereby, the IPv6 interface identifier is "0000:DE00:00A2:58A5", considering the documentation network 2001:db8::/32. The final IPv6 address for the legacy device is "2001:db8::DE00:A2:58A5".

The address is presented in the Figure 2.

Tech. ID MSB (6 bits) 0x00	U/L "0" (1 bit) 0	Tech. ID LSB (5 bit) 0x00	Reserved (4 bits) 0x00	Mapping MSB (8 bits) 0xDE	EUI-64 "0x0000" (16 bits) 0x0000	Mapping LSBs (24 bits) 0xA258A5
-------------------------------------	----------------------------	------------------------------------	------------------------------	------------------------------------	---	--

Figure 2: EIB/KNX example: the IPv6 interface identifier.

6.2. Example 2 - RFID

We assume the legacy protocol is RFID. Each RFID device is identified by its Electronic Product Code (EPC), whose length may vary from 96 to 256 bits. Let us assume to have an RFID device whose EPC is given by 01.23F3D00.8666A3.000000A05 (12 bytes). Let us assume that the RFID technology ID is "1".

We apply a hash (CRC-32) to the sequence 0x0123F3D008666A3000000A05. The result is 0xA93AFFA0.

Thereby, the IPv6 interface identifier is "0004:A900:003A:FFA0", considering the documentation network 2001:db8::/32. The final IPv6 address for the RFID tag is "2001:db8::400:A900:3A:FFA0".

The address is presented in the Figure 2.

Tech. ID MSB (6 bits) 0x00	U/L "0" (1 bit) 0	Tech. ID (5 bit) 0x04	Reserved (4 bits) 0x00	Mapping MSB (8 bits) 0xA9	EUI-64 "0x0000" (16 bits) 0x0000	Mapping LSBs (24 bits) 0x3AFFA0
-------------------------------------	----------------------------	--------------------------------	------------------------------	------------------------------------	---	--

Figure 3: RFID example: the IPv6 interface identifier.

7. IANA Considerations

Not yet defined.

8. Security considerations

The proposed mapping mechanism, being based on mapping proprietary protocol ID, results in such ID being incorporated in the final IPv6 address, exposing this piece of information to the Internet. The concern has been that a user might not want to expose the details of the system to outsiders. For such concern, which holds also for MAC address mapping into EUI64 addresses, please refer to appendix B in [RFC4942].

9. Acknowledgements

The authors wish to acknowledge the following for their review and constructive criticism of this proposal: Robert Cragie. Thanks to the IoT6 European Project (STREP) of the 7th Framework Program (Grant 288445), and the colleagues who have collaborated in this work. In particular, Antonio Skarmeta from the University of Murcia, Peter Kirstein and Socrates Varakliotis from the University College London, and Sebastien Ziegler from Mandat International.

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [SENSORS] Jara, A., Moreno-Sanchez, P., Skarmeta, A., Varakliotis, S., and P. Kirstein,, "IPv6 Addressing Proxy: Mapping Native Addressing from Legacy Technologies and Devices to the Internet of Things (IPv6)", Sensors 13, no. 5, 6687-6712, 2013, 2013.

Authors' Addresses

Gianluca Rizzo, Ed.
HES-SO Valais
Technopole 3
Sierre, Valais 3960
Switzerland

Phone: +41-76-6151758
Email: gianluca.rizzo@hevs.ch

Antonio J. Jara, Ed.
HES-SO Valais
Technopole 3
Sierre, Valais 3960
Switzerland

Email: jara@ieee.org

Alex C. Olivieri
HES-SO Valais
Technopole 3
Sierre, Valais 3960
Switzerland

Email: Alex.Olivieri@hevs.ch

Yann Bocchi
HES-SO Valais
Technopole 3
Sierre, Valais 3960
Switzerland

Email: yann.bocchi@hevs.ch

Maria Rita Palattella
University of Luxembourg
4, rue Alphonse Weicker
Interdisciplinary Centre for Security, Reliability and Trust
Luxembourg

Phone: (+352) 46 66 44 5841
Email: maria-rita.palattella@uni.lu

Latif Ladid
University of Luxembourg / IPv6 Forum
4, rue Alphonse Weicker
Interdisciplinary Centre for Security, Reliability and Trust
Luxembourg

Phone: (+352) 46 66 44 5720
Email: latif@ladid.lu

Sebastien Ziegler
Mandat International
3 rue Champ Baron
1209 Geneva
Switzerland

Email: sziegler@mandint.org

Cedric Crettaz
Mandat International
3 rue Champ Baron
1209 Geneva
Switzerland

Email: iot6@mandint.org

6Lo Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 4, 2014

T. Savolainen
Nokia
January 31, 2014

Optimal Transmission Window Option for ICMPv6 Router Advertisement
draft-savolainen-6lo-optimal-transmission-window-00

Abstract

For a class of gateways an activation of an uplink network connection, such as cellular radio connection, incurs a fixed cost in form of consumed energy. For these gateways minimizing the number of uplink activations is of importance. This specification describes an Optimal Transmission Window option for ICMPv6 Router Advertisement that a gateway can use to communicate optimal transmission window for nodes it is serving, thus helping to group separate transmissions together and thereby reduce number of gateway's uplink connection activation events.

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 August 4, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Problem Description	3
3. Solution Description	4
4. Optimal Transmission Window Option	5
5. Gateway Behavior	5
6. Node Behavior	6
7. Protocol Constants	7
8. Acknowledgements	8
9. Contributors	8
10. IANA Considerations	8
11. Security Considerations	8
12. References	8
12.1. Normative References	8
12.2. Informative References	9
Author's Address	10

1. Introduction

In certain deployments gateways are very power constrained. A class of such gateways are battery powered cellular-using gateways that are sharing the wireless cellular connection to other nodes in wireless local area networks (LAN) such as IEEE 802.11, 802.15.4, or Bluetooth Low-Energy networks. Hosts in LANs may be, for example, personal computers, tablets, low-energy sensors and actuators, and alike.

Use of the cellular uplink contributes significant power consumption for the gateway device, which provides motivation for minimizing time and frequency of cellular uplink usage. The causes for power consumption are discussed further in Section 2.

This document describes an ICMPv6 Router Advertisement [RFC4861] Optimal Transmission Window option, which a gateway can use in an attempt to schedule and synchronize periodical communication activities of the nodes gateway provides forwarding services for. The option describes an optimal transmission window, during which nodes should perform periodic and time insensitive transmissions. This helps to minize the power consumption of the gateway by reducing numbers of cellular radio activation events.

1.1. Requirements Language

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

2. Problem Description

3GPP cellular networks require establishment of a Packed Data Protocol (PDP) context or Evolved Packet System (EPS) bearer in order to transmit IP packets [RFC6459]. This establishment consumes energy, but for long-lived connections, such as always-on connections, the relative significance is small. The established cellular connection can then be shared to LAN by using DHCPv6 Prefix Delegation [RFC6459], by extending an IPv6 /64 prefix of the cellular connection [I-D.ietf-v6ops-64share], or by utilizing Network Address Translation (NAT) techniques.

In order to save power and bandwidth, 3GPP cellular radio attempts to enter and stay in idle mode whenever there is nothing to transmit. During this idle mode the logical IP-connection is retained.

Whenever data needs to be transmitted over 3GPP radio connection that is currently in idle mode, the connection has to be signaled active. Once the connection is active, actual user data can be transmitted. After the user data has been transmitted, the 3GPP connection is kept active for operator configurable time waiting for possible additional data. If no additional data is transmitted within this time, the radio is returned back to the idle mode.

Total energy consumed for a transmission event consist of signaling required for activation of radio, transmission of the actual user data, keeping the radio active while waiting for possible additional data to be transmitted, and deactivation of the radio. Balasubramanian et al. refer with 'ramp energy' for energy spent on the radio activation and with 'tail energy' for the energy spent after the transmission of the user data [Balasubramanian2009].

The exact features of 3GPP radio for which the energy is consumed varies by the network generation. In 2G GPRS and EDGE networks setup and teardown of Temporary Block Flows (TBF) are responsible of big part of 'tail energy'. In 3G WCDMA, HSPA, transitions through Radio Resource Control (RRC) states before returning to idle are sources of 'tail energy' consumption [Haverinen2007]. In 4G LTE networks support and parameters of discontinuous reception (DRX) affect significantly on the power consumption [Bontu2009] .

While 3GPP cellular network technologies are used as an example herein, other radio technologies may have similar properties causing 'ramp energy' and 'tail energy' consumption.

In case of small transactions, such as with keep-alive messaging or resource state updates, the 'ramp energy' and 'tail energy' contribute very significant part of the total energy consumption. For single device use-cases this is the state of art, and there is not much that can be optimized.

However, in the scenario where a cellular-using gateway is serving multitude of devices in LAN, it can happen that significant energy is unnecessarily spent on 'ramp energy' and 'tail energy'. This happens when multiple devices in LAN are transmitting data seldomly and with such intervals that the cellular gateway has to separately activate the cellular radio for each transaction. I.e. in scenario where each transaction from devices in LAN causes 'ramp energy' and 'tail energy' costs for the gateway.

Hence the problem is: how to optimize energy spent on 'ramp energy' and 'tail energy' in case of battery powered cellular-using gateway serving multitude of devices in LAN.

3. Solution Description

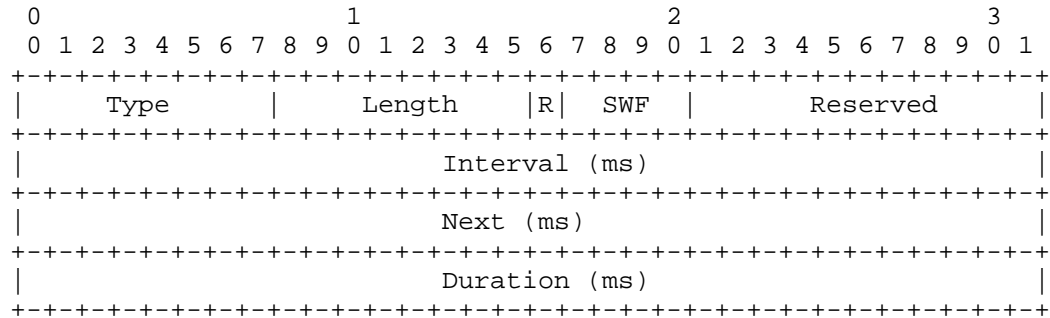
To solve the problem described in Section 2, this document presents a method for a gateway to attempt grouping of seldom and periodic communications performed by nodes in the LAN. The solution does not help in power consumption caused by transmissions initiated from the Internet.

The gateway performs transmission grouping by indicating to nodes in LAN optimal transmission window using option defined in Section 4. The nodes will then attempt to send data that is not time critical at the optimal times indicated by the gateway. This can work, for example, when nodes need to perform periodic keep-alive signaling, periodically poll or push data, or for example are using CoAP observe [I-D.ietf-core-observe] and need to send resource state updates that are not time critical.

The algorithms and procedures for nodes to switch to utilize optimal transmission window, and the algorithms and procedures for the gateway to come up with interval and duration parameter values for optimal transmission window, are left for implementations to choose.

4. Optimal Transmission Window Option

The Optimal Transmission Window is communicated from gateway to the nodes by including Optimal Transmission Window Option within ICMPv6 Router Advertisements [RFC4861]. The option is defined below.



Type: TBD

Length: 2

R: If set, the optimal transmission window is open when the Router Advertisement was sent. If not set, the window may not be open.

SWF: Decimal value indicating secondary transmission window timing as fractions of Interval. Value of zero indicates lack of secondary transmission windows. Other values are used as dividers for Interval. Default value is decimal 8 (binary '1000').

Reserved: Reserved for the future, MUST be set to zero.

Interval: The time between optimal transmission windows, in milliseconds.

Next: The time to the start of the next optimal transmission window in milliseconds.

Duration: The time the optimal transmission window is open in milliseconds, for example, how long the gateway estimates the radio to be at least active.

5. Gateway Behavior

A gateway that attempts to synchronize periodic transmission of nodes it serves SHOULD include Optimal Transmission Window option in all ICMPv6 Router Advertisement messages it originates.

If the uplink radio is active at the time of sending the Router Advertisement, the gateway SHOULD set the R-bit on to indicate immediately suitable time for transmissions. Furthermore, in the event of uplink radio activation, a gateway MAY send otherwise

unscheduled Router Advertisement message with R-bit set in order to indicate unscheduled power efficient transmission opportunity for nodes.

The gateway using this option MUST set the Interval-field to exactly match the optimal sending window, as some nodes receiving the ICMPv6 Router Advertisement can choose to go to sleep until the optimal transmission window opens. The value for the interval-field is gateway's implementation decision and depends on the deployment scenario. A default value of INTERVAL_DEFAULT (see Section 7) is defined for the cases where gateway has no better information. Interval field value of zero indicates transmission window to be always open. The SWF-field indicates presence and time of secondary transmission windows during one Interval. For example, default value of 8 indicates secondary transmission window to occur at every INTERVAL_DEFAULT/8.

With the default values for INTERVAL_DEFAULT and SWF-field nodes have secondary transmission window every 100 seconds, which is enough in case host needs to refresh UDP mappings of NAT utilizing two minute expiration time (see section 4.3 of [RFC4787]).

The Next-field MUST be always set to point to the moment of the next optimal transmission window. Even if the R-bit is set, the Next-field MUST nevertheless point to the start of the next optimal transmission window.

The Duration-field MUST indicate the length of the window during which hosts should start their periodic transmissions. The length has to be at least MIN_WINDOW_DURATION (see Section 7).

The secondary transmission window bitfield indicates possibly alternative, but still synchronized, times for hosts to transmit if the optimal sending window interval frequency is too low.

If the gateway implements synchronization services for gateway's internal applications' periodical communications, the gateway MUST synchronize the internal applications to communicate during the same optimal transmission window.

6. Node Behavior

A node implementing this specification SHOULD utilize the timing information received via Optimal Transmission Window option and time it's periodic transmissions accordingly when possible. Additionally, a node MAY use Router Advertisement with this option and R-bit set as trigger for communications. The node MUST refresh it's timing states

after every received Router Advertisement message having the Optimal Transmission Window option.

The node **MUST** wait for a random period of time between the start of the optimal transmission window, or reception of a Router Advertisement with R-bit set, and COLLISION_AVOIDANCE_DURATION (see Section 7) in order to avoid collisions caused by multitude of nodes transmitting at the same time.

Sometimes a node needs to perform time consuming operations on the link before transmitting to the Internet, such as performing Detecting Network Attachment-procedures [RFC6059] if the node has been asleep long enough. In such cases, the node **SHOULD** perform time consuming operations before the communications are scheduled to take place.

The node does not have to transmit during every window, but **SHOULD** use the one right before the application's optimal periodic communication event would occur. If the node is running application that requires more frequent periodic messaging than what the optimal transmission window provides, the host **SHOULD** attempt to communicate during secondary transmission windows as configured via SWF-field.

The node **MUST** only use timing values as learned from the Router Advertisement message that has been used for the highest priority default router configuration. If the node supports more-specific routes [RFC4191], the node **SHOULD** also record optimal transmission window schedules for each more-specific route.

The node **SHOULD** provide an implementation specific application programming interface that applications can use to learn the optimal transmission window schedules. If the node maintains destination-specific optimal transmission window timing information, the application programming interface **SHOULD** allow applications to ask for the timing information specific to a destination.

7. Protocol Constants

Following constants are defined for the operation of the Optimal Transmission Window option.

INTERVAL_DEFAULT: 800 seconds

MIN_WINDOW_DURATION: 500 milliseconds

COLLISION_AVOIDANCE_DURATION: 100 milliseconds

8. Acknowledgements

We ack.

9. Contributors

Johanna Nieminen contributed to and was the second author of the first IETF contribution of this problem and solution (draft-savolainen-6man-optimal-transmission-window-00).

10. IANA Considerations

This memo requests IANA to register a new Neighbor Discovery Option Type under the subregistry "IPv6 Neighbor Discovery Option Formats" of the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry ([http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml](http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters/icmpv6-parameters.xhtml)).

The type number can be the next available.

The Description would be: "Optimal Transmission Window Option".

11. Security Considerations

This document specifies that a node uses timing information only from the Router Advertisements the node accepts for configuring default and more-specific routes. This helps to mitigate against attacks that try to influence transmission schedules by sending malicious Router Advertisements.

With this option it is not possible to hinder node's communications, as the option is a power saving optimization that help nodes to synchronize transmissions with each other, while still allowing transmissions at any time when necessary. Therefore, if the timing values sent in Router Advertisement do not make sense for a node, or it's applications, the values can be ignored.

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.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

12.2. Informative References

- [Balasubramanian2009]
Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications Energy Consumption of Always-On Applications in WCDMA Networks", November 2009.
- [Bontu2009]
Chandra Sekhar Bontu and Ed Illidge, "DRX Mechanism for Power Saving in LTE", June 2009.
- [Haverinen2007]
Henry Haverinen, Jonne Siren, and Pasi Eronen, "Energy Consumption of Always-On Applications in WCDMA Networks", April 2007.
- [I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-11 (work in progress), October 2013.
- [I-D.ietf-v6ops-64share]
Byrne, C., Drown, D., and V. Ales, "Extending an IPv6 /64 Prefix from a 3GPP Mobile Interface to a LAN link", draft-ietf-v6ops-64share-09 (work in progress), October 2013.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, November 2010.
- [RFC6459] Korhonen, J., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, January 2012.

Author's Address

Teemu Savolainen
Nokia
Visiokatu 3
Tampere FI-33720
Finland

Email: teemu.savolainen@nokia.com