

lpwan
Internet-Draft
Intended status: Informational
Expires: May 1, 2017

S. Farrell
Trinity College Dublin
A. Yegin
Actility
October 28, 2016

LoRaWAN Overview
draft-farrell-lpwan-lora-overview-01

Abstract

Low Power Wide Area Networks (LPWAN) are wireless technologies covering different Internet of Things (IoT) applications. The common characteristics for LPWANs are large coverage, low bandwidth, small packet and application layer data sizes and long battery life operation. One of these technologies is LoRaWAN developed by the LoRa Alliance. LoRaWAN targets key requirements of the Internet of things such as secure bi-directional communication, mobility and localization services. This memo is an informational overview of LoRaWAN and gives the principal characteristics of this technology in order to help with the IETF work for providing IPv6 connectivity over LoRaWAN along with other LPWANs.

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 1, 2017.

Copyright Notice

Copyright (c) 2016 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. Terminology	3
3. Radio Spectrum	4
4. MAC Layer	6
5. Names and Addressing	8
6. Security Considerations	10
6.1. Payload Encryption and Data Integrity	10
6.2. Key Derivation	10
7. IANA Considerations	11
8. Acknowledgements	11
9. Contributors	11
10. Informative References	12
Authors' Addresses	12

1. Introduction

LoRaWAN is a wireless technology for long-range low-power low-data-rate applications developed by the LoRa Alliance, a membership consortium. [<https://www.lora-alliance.org/>](https://www.lora-alliance.org/) LoRaWAN networks are typically organized in a star-of-stars topology in which gateways relay messages between end-devices and a central "network server" in the backend. Gateways are connected to the network server via IP links while end-devices use single-hop LoRaWAN communication that can be received at one or more gateways. All communication is generally bi-directional, although uplink communication from end-devices to the network server are favoured in terms of overall bandwidth availability.

In LoRaWAN networks, end-device transmissions may be received at multiple gateways, so during nominal operation a network server may see multiple instances of the same uplink message from an end-device.

To maximize both battery life of end-devices and overall network capacity, the LoRaWAN network infrastructure manages the data rate and RF output power for each end-device individually by means of an adaptive data rate (ADR) scheme. End-devices may transmit on any channel allowed by local regulation at any time, using any of the currently available data rates.

This memo provides an overview of the LoRaWAN technology for the Internet community, but the definitive specification [LoRaSpec] is that produced by the LoRa Alliance. This draft is based on version 1.0.2 of the LoRa specification. (Note that version 1.0.2 is expected to be published in a few weeks. We will update this draft when that has happened. For now, version 1.0 is available at [LoRaSpec1.0])

2. Terminology

This section introduces some LoRaWAN terms. Figure 1 shows the entities involved in a LoRaWAN network.

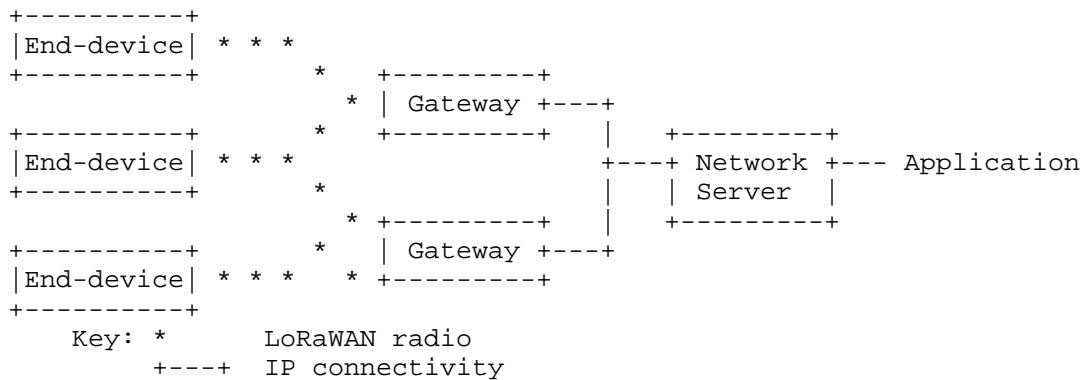


Figure 1: LoRaWAN architecture

- o End-device: a LoRa client device, sometimes called a mote. Communicates with gateways.
- o Gateway: a radio on the infrastructure-side, sometimes called a concentrator or base-station. Communicates with end-devices and, via IP, with a network server.
- o Network Server: The Network Server (NS) terminates the LoRaWAN MAC layer for the end-devices connected to the network. It is the center of the star topology.
- o Uplink message: refers to communications from end-device to network server or application via one or more gateways.
- o Downlink message: refers to communications from network server or application via one gateway to a single end-device or a group of end-devices (considering multicasting).

- o Application: refers to application layer code both on the end-device and running "behind" the network server. For LoRaWAN, there will generally only be one application running on most end-devices. Interfaces between the network server and application are not further described here.
- o Classes A, B and C define different device capabilities and modes of operation for end-devices. End-devices can transmit uplink messages at any time in any mode of operation (so long as e.g., ISM band restrictions are honoured). An end-device in Class A can only receive downlink messages at predetermined timeslots after each uplink message transmission. Class B allows the end-device to receive downlink messages at periodically scheduled timeslots. Class C allows receipt of downlink messages at anytime. Class selection is based on the end-devices' application use case and its power supply. (While Classes B and C are not further described here, readers may have seen those terms elsewhere so we include them for clarity.)

3. Radio Spectrum

LoRaWAN radios make use of ISM bands, for example, 433MHz and 868MHz within the European Union and 915MHz in the Americas.

The end-device changes channel in a pseudo-random fashion for every transmission to help make the system more robust to interference and/or to conform to local regulations.

As with other LPWAN radio technologies, LoRaWAN end-devices respect the frequency, power and maximum transmit duty cycle requirements for the sub-band imposed by local regulators. In most cases, this means an end-device is only transmitting for 1% of the time, as specified by ISM band regulations. And in some cases the LoRaWAN specification calls for end-devices to transmit less often than is called for by the ISM band regulations in order to avoid congestion.

Figure 2 below shows that after a transmission slot a Class A device turns on its receiver for two short receive windows that are offset from the end of the transmission window. The frequencies and data rate chosen for the first of these receive windows match those used for the transmit window. The frequency and data-rate for the second receive window are configurable. If a downlink message preamble is detected during a receive window, then the end-device keeps the radio on in order to receive the frame.

End-devices can only transmit a subsequent uplink frame after the end of the associated receive windows. When a device joins a LoRaWAN

network (see Section 4 for details), there are similar timeouts on parts of that process.

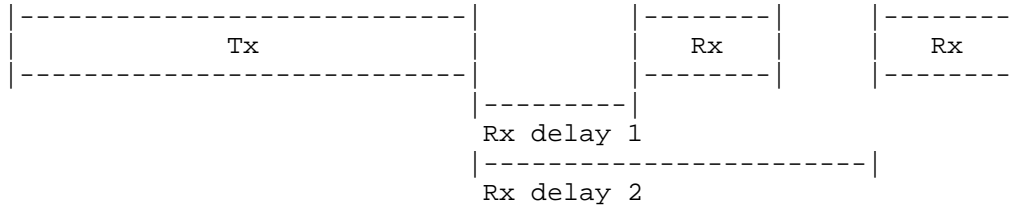


Figure 2: LoRaWAN Class A transmission and reception window

Given the different regional requirements the detailed specification for the LoRaWAN physical layer (taking up more than 30 pages of the specification) is not reproduced here. Instead and mainly to illustrate the kinds of issue encountered, in Table 1 we present some of the default settings for one ISM band (without fully explaining those here) and in Table 2 we describe maxima and minima for some parameters of interest to those defining ways to use IETF protocols over the LoRaWAN MAC layer.

Parameters	Default Value
Rx delay 1	1 s
Rx delay 2	2 s (must be RECEIVE_DELAY1 + 1s)
join delay 1	5 s
join delay 2	6 s
868MHz Default channels	3 (868.1,868.2,868.3), data rate: 0.3-5 kbps

Table 1: Default settings for EU868MHz band

Parameter/Notes	Min	Max
Duty Cycle: some but not all ISM bands impose a limit in terms of how often an end-device can transmit. In some cases LoRaWAN is more stringent in an attempt to avoid congestion.	1%	no-limit
EU 868MHz band data rate/frame-size	250 bits/s : 59 octets	50000 bits/s : 250 octets
US 915MHz band data rate/frame-size	980 bits/s : 19 octets	21900 bits/s : 250 octets

Table 2: Minima and Maxima for various LoRaWAN Parameters

Note that in the case of the smallest frame size (19 octets), 8 octets are required for LoRa MAC layer headers leaving only 11 octets for payload (including MAC layer options). However, those settings do not apply for the join procedure - end-devices are required to use a channel that can send the 23 byte Join-request message for the join procedure.

4. MAC Layer

Uplink and downlink higher layer data is carried in a MACPayload. There is a concept of "ports" (an optional 8 bit value) to handle different applications on an end-device. Port zero is reserved for LoRaWAN specific messaging, such as the join procedure.

The header also distinguishes the uplink/downlink directions and whether or not an acknowledgement ("confirmation") is required from the peer.

All payloads are encrypted and ciphertxts are protected with a cryptographic Message Integrity Check (MIC) - see Section 6 for details.

In addition to carrying higher layer PDUs there are Join-Request and Join-Response (aka Join-Accept) messages for handling network access. And so-called "MAC commands" (see below) up to 15 bytes long can be piggybacked in an options field ("FOpts").

LoRaWAN end-devices can choose various different data rates from a menu of available rates (dependent on the frequencies in use). It is however, recommended that end-devices set the Adaptive Data Rate ("ADR") bit in the MAC layer which is a signal that the network should control the data rate (via MAC commands to the end-device). The network can also assert the ADR bit and control data rates at its discretion. The goal is to ensure minimal on-time for radios whilst increasing throughput and reliability when possible. Other things being equal, the effect should be that end-devices closer to a gateway can successfully use higher data rates, whereas end-devices further from all gateways still receive connectivity though at a lower data rate.

Data rate changes can be validated via a scheme of acks from the network with a fall-back to lower rates in the event that downlink acks go missing.

There are 16 (or 32) bit frame counters maintained in each direction that are incremented on each transmission (but not re-transmissions) that are not re-used for a given key. When the device supports a 32 bit counter, then only the least significant 16 bits are sent in the MAC header, but all 32 bits are used in cryptographic operations. (If an end-device only supports a 16 bit counter internally, then the topmost 16 bits are set to zero.)

There are a number of MAC commands for: Link and device status checking, ADR and duty-cycle negotiation, managing the RX windows and radio channel settings. For example, the link check response message allows the network server (in response to a request from an end-device) to inform an end-device about the signal attenuation seen most recently at a gateway, and to also tell the end-device how many gateways received the corresponding link request MAC command.

Some MAC commands are initiated by the network server. For example, one command allows the network server to ask an end-device to reduce its duty-cycle to only use a proportion of the maximum allowed in a region. Another allows the network server to query the end-device's power status with the response from the end-device specifying whether it has an external power source or is battery powered (in which case a relative battery level is also sent to the network server).

The network server can also inform an end-device about channel assignments (mid-point frequencies and data rates). Of course, these must also remain within the bands assigned by local regulation.

5. Names and Addressing

A LoRaWAN network has a short network identifier ("NwkID") which is a seven bit value. A private network (common for LoRaWAN) can use the value zero. If a network wishes to support "foreign" end-devices then the NwkID needs to be registered with the LoRA Alliance, in which case the NwkID is the seven least significant bits of a registered 24-bit NetID. (Note however, that the methods for "roaming" are currently being enhanced within the LoRA Alliance, so the situation here is somewhat fluid.)

In order to operate nominally on a LoRaWAN network, a device needs a 32-bit device address, which is the concatenation of the NwkID and a 25-bit device-specific network address that is assigned when the device "joins" the network (see below for the join procedure) or that is pre-provisioned into the device.

End-devices are assumed to work with one or a quite limited number of applications, which matches most LoRaWAN use-cases. The applications are identified by a 64-bit AppEUI, which is assumed to be a registered IEEE EUI64 value.

In addition, a device needs to have two symmetric session keys, one for protecting network artefacts (port=0), the NwkSKey, and another for protecting application layer traffic, the AppSKey. Both keys are used for 128 bit AES cryptographic operations. (See Section 6 for details.)

So, one option is for an end-device to have all of the above, plus channel information, somehow (pre-)provisioned, in which case the end-device can simply start transmitting. This is achievable in many cases via out-of-band means given the nature of LoRaWAN networks. Table 3 summarises these values.

Value	Description
DevAddr	DevAddr (32-bits) = NwkId (7-bits) + device-specific network address (25 bits)
AppEUI	IEEE EUI64 naming the application
NwkSKey	128 bit network session key for use with AES
AppSKey	128 bit application session key for use with AES

Table 3: Values required for nominal operation

As an alternative, end-devices can use the LoRaWAN join procedure in order to setup some of these values and dynamically gain access to the network.

To use the join procedure, an end-device must still know the AppEUI. In addition to the AppEUI, end-devices using the join procedure need to also know a different (long-term) symmetric key that is bound to the AppEUI - this is the application key (AppKey), and is distinct from the application session key (AppSKey). The AppKey is required to be specific to the device, that is, each end-device should have a different AppKey value. And finally the end-device also needs a long-term identifier for itself, syntactically also an EUI-64, and known as the device EUI or DevEUI. Table 4 summarises these values.

Value	Description
DevEUI	IEEE EUI64 naming the device
AppEUI	IEEE EUI64 naming the application
AppKey	128 bit long term application key for use with AES

Table 4: Values required for join procedure

The join procedure involves a special exchange where the end-device asserts the AppEUI and DevEUI (integrity protected with the long-term AppKey, but not encrypted) in a Join-request uplink message. This is then routed to the network server which interacts with an entity that knows that AppKey to verify the Join-request. All going well, a Join-accept downlink message is returned from the network server to the end-device that specifies the 24-bit NetID, 32-bit DevAddr and channel information and from which the AppSKey and NwkSKey can be derived based on knowledge of the AppKey. This provides the end-device with all the values listed in Table 3.

There is some special handling related to which channels to use and for multiple transmissions for the join-request which is intended to ensure a successful join in as many cases as possible. Join-request and Join-accept messages also include some random values (nonces) to both provide some replay protection and to help ensure the session keys are unique per run of the join procedure. If a Join-request fails validation, then no Join-accept message (indeed no message at all) is returned to the end-device. For example, if an end-device is factory-reset then it should end up in a state in which it can re-do the join procedure.

6. Security Considerations

In this section we describe the use of cryptography in LoRaWAN. This section is not intended as a full specification but to be sufficient so that future IETF specifications can encompass the required security considerations. The emphasis is on describing the externally visible characteristics of LoRaWAN.

6.1. Payload Encryption and Data Integrity

All payloads are encrypted and have data integrity. Frame options (used for MAC commands) when sent as a payload (port zero) are therefore protected. MAC commands piggy-backed as frame options ("FOpts") are however sent in clear. Since MAC commands may be sent as options and not only as payload, any values sent in that manner are visible to a passive attacker but are not malleable for an active attacker due to the use of the MIC.

For LoRaWAN version 1.0.x, the NWkSKey session key is used to provide data integrity between the end-device and the network server. The AppSKey is used to provide data confidentiality between the end-device and network server, or to the application "behind" the network server, depending on the implementation of the network.

All MAC layer messages have an outer 32-bit Message Integrity Code (MIC) calculated using AES-CMAC calculated over the ciphertext payload and other headers and using the NwkSKey.

Payloads are encrypted using AES-128, with a counter-mode derived from IEEE 802.15.4 using the AppSKey.

Gateways are not expected to be provided with the AppSKey or NwkSKey, all of the infrastructure-side cryptography happens in (or "behind") the network server.

6.2. Key Derivation

When session keys are derived from the AppKey as a result of the join procedure the Join-accept message payload is specially handled.

The long-term AppKey is directly used to protect the Join-accept message content, but the function used is not an aes-encrypt operation, but rather an aes-decrypt operation. The justification is that this means that the end-device only needs to implement the aes-encrypt operation. (The counter mode variant used for payload decryption means the end-device doesn't need an aes-decrypt primitive.)

The Join-accept plaintext is always less than 16 bytes long, so electronic code book (ECB) mode is used for protecting Join-accept messages.

The Join-accept contains an AppNonce (a 24 bit value) that is recovered on the end-device along with the other Join-accept content (e.g. DevAddr) using the aes-encrypt operation.

Once the Join-accept payload is available to the end-device the session keys are derived from the AppKey, AppNonce and other values, again using an ECB mode aes-encrypt operation, with the plaintext input being a maximum of 16 octets.

7. IANA Considerations

There are no IANA considerations related to this memo.

8. Acknowledgements

The authors re-used some text from [I-D.vilajosana-lpwan-lora-hc]

Stephen Farrell's work on this memo was supported by the Science Foundation Ireleand funded CONNECT centre <<https://connectcentre.ie/>>.

9. Contributors

The following members of the LoRa Alliance reviewed this draft and contributed (much more than SF) to the definition of LoRaWAN.

Name, Affiliation, email (optional)

Chun-Yeow Yeoh, VADS LYFE SDN BHD, yeow@tmrnd.com.my

Olivier Hersent, Actility, olivier.hersent@actility.com

Dave Kjendal, Senet Inc, dkjendal@senetco.com

Paul Duffy, Cisco, paduffy@cisco.com

Joachim Ernst, Swisscom Broadcast Ltd, joachim.ernst@swisscom.com

Nicolas Sornin, Semtech, nsornin@semtech.com

Phillippe Christin, Orange, philippe.christin@orange.com

10. Informative References

[I-D.vilajosana-lpwan-lora-hc]

Vilajosana, X., Dohler, M., and A. Yegin, "Transmission of IPv6 Packets over LoRaWAN", draft-vilajosana-lpwan-lora-hc-00 (work in progress), July 2016.

[LoRaSpec]

LoRa Alliance, "LoRaWAN Specification Version V1.0.2", Nov 2016, <URL TBD>.

[LoRaSpec1.0]

LoRa Alliance, "LoRaWAN Specification Version V1.0", Jan 2015, <<https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>>.

Authors' Addresses

Stephen Farrell
Trinity College Dublin
Dublin 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie

Alper Yegin
Actility
Paris, Paris
FR

Email: alper.yegin@actility.com

lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: March 10, 2018

A. Minaburo
Acklio
L. Toutain
Institut MINES TELECOM ; IMT Atlantique
September 06, 2017

LPWAN Static Context Header Compression (SCHC) for CoAP
draft-ietf-lpwan-coap-static-context-hc-02

Abstract

This draft defines the way SCHC header compression can be applied to CoAP headers. CoAP header structure differs from IPv6 and UDP protocols since the CoAP Header is flexible header with a variable number of options themselves of a variable length. Another important difference is the asymmetry in the header information used for request and response messages. This draft takes into account the fact that a thing can play the role of a CoAP client, a CoAP client or both roles.

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 <https://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 10, 2018.

Copyright Notice

Copyright (c) 2017 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 (<https://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. CoAP Compressing	3
3. Compression of CoAP header fields	4
3.1. CoAP version field (2 bits)	4
3.2. CoAP type field	5
3.3. CoAP token length field	5
3.4. CoAP code field	6
3.5. CoAP Message ID field	8
3.6. CoAP Token field	9
4. CoAP options	9
4.1. CoAP option Content-format field.	9
4.2. CoAP option Accept field	10
4.3. CoAP option Max-Age field, CoAP option Uri-Host and Uri-Port fields	11
5. CoAP option Uri-Path and Uri-Query fields	11
5.1. CoAP option Proxy-URI and Proxy-Scheme fields	12
5.2. CoAP option ETag, If-Match, If-None-Match, Location-Path and Location-Query fields	13
6. Other RFCs	13
6.1. Block	13
6.2. Observe	13
6.3. No-Response	13
7. Protocol analysis	13
8. Examples of CoAP header compression	14
8.1. Mandatory header with CON message	14
8.2. Complete exchange	16
9. Normative References	17
Authors' Addresses	18

1. Introduction

CoAP [rfc7252] is an implementation of the REST architecture for constrained devices. Gateway between CoAP and HTTP can be easily built since both protocols uses the same address space (URL), caching mechanisms and methods.

Nevertheless, if limited, the size of a CoAP header may be too large for LPWAN constraints and some compression may be needed to reduce the header size.

[I-D.toutain-lpwan-ipv6-static-context-hc] defines a header compression mechanism for LPWAN network based on a static context. The context is said static since the element values composing the context are not learned during the packet exchanges but are previously defined. The context(s) is(are) known by both ends before transmission.

A context is composed of a set of rules (contexts) that are referenced by Rule IDs (identifiers). A rule contains an ordered list of the header fields containing a field ID (FID) and its position when repeated, a direction indicator (DI) (upstream, downstream and bidirectional) and some associated Target Values (TV) which are expected in the message header. A Matching Operator (MO) is associated to each header field description. The rule is selected if all the MOs fit the TVs. In that case, a Compression Decompression Function (CDF) associated to each field defines the link between the compressed and decompressed value for each of the header fields.

This document describes how the rules can be applied to CoAP flows. Compression of the CoAP header may be done in conjunction with the above layers or independantly.

2. CoAP Compressing

CoAP differs from IPv6 and UDP protocols on the following aspects:

- o IPv6 and UDP are symmetrical protocols. The same fields are found in the request and in the response, only location in the header may vary (e.g. source and destination fields). A CoAP request is different from an response. For example, the URI-path option is mandatory in the request and is not found in the response, request may contain an Accept option and the response a Content-format option.

Even when a field is "symmetric" (i.e. found in both directions) the values carried are different. For instance the Type field will contain a CON value in the request and a ACK or RST value in the response. Exploiting the asymmetry in compression will allow to send no bit in the compressed request and a single bit in the answer. Same behavior can be applied to the CoAP Code field (0.OX code are present in the request and Y.ZZ in the answer).

- o CoAP also obeys to the client/server paradigm and the compression rate can be different if the request is issued from a LPWAN node or from an non LPWAN device. For instance a Thing (ES) aware of LPWAN constraints can generate a 1 byte token, but a regular CoAP client will certainly send a larger token to the Thing. SCHC

compression will not modify the values to offer a better compression rate. Nevertheless a proxy placed before the compressor may change some field values to offer a better compression rate and maintain the necessary context for interoperability with existing CoAP implementations.

- o In IPv6 and UDP header fields have a fixed size. In CoAP, Token size may vary from 0 to 8 bytes, length is given by a field in the header. More systematically, the CoAP options are described using the Type-Length-Value. When applying SCHC header compression.

By sending compressed field information following the rule order, SCHC offers a serialization/deserialization mechanism. Since a field exists to indicate the token length there is no ambiguity. For options, the rule indicates also the expected options found the int CoAP header. Therefore only the length is needed to recognise an option. The length will be send using the same CoAP encoding (size less than 12 are directly sent, higher values uses the escape mechanisms defined by [rfc7252]). Delta Type is omitted, the value will be recovered by the decompressor. This reduce the option length of 4, 12 or 20 bits regarding the original size of the delta type encoding in the option.

- o In CoAP headers a field can be duplicated several times, for instances, elements of an URI (path or queries) or accepted formats. The position defined in a rule, associated to a Field ID, can be used to identify the proper element.

3. Compression of CoAP header fields

This section discusses of the compression of the different CoAP header fields. These are just examples. The compression should take into account the nature of the traffic and not only the field values. Next chapter will define some compression rules for some common exchanges.

3.1. CoAP version field (2 bits)

This field is bidirectional and can be elided during the SCHC compression, since it always contains the same value. It appears only in first position.

FID	Pos	DI	TV	MO	CDF
ver	1	bi	1	equal	not-sent

3.2. CoAP type field

This field can be managed bidirectionally or unidirectionally. Several strategies can be applied to this field regarding the values used:

- o If the ES is a client or a Server and non confirmable message are used, the transmission of the Type field can be avoided:

- * Pos is always 1,
- * DI can either be "uplink" if the ES is a CoAP client or "downlink" if the ES is a CoAP server, or "bidirectional"
- * TV is set to the value,
- * MO is set to "equal"
- * CDF is set to "not-sent".

FID	Pos	DI	TV	MO	CDF
type	1	bi	NON	equal	not-sent

- o If the ES is either a client or a Server and confirmable message are used, the DI can be used to elide the type on the request and compress it to 1 bit on the response. The example above shows the rule for a ES acting as a client, directions need to be reversed for a ES acting as a server.

FID	Pos	DI	TV	MO	CDF
type	1	up	CON	equal	not-sent
type	1	dw	{0:ACK, 1:RST}	match-mapping	mapping-sent

- o Otherwise if the ES is acting simultaneously as a client and a server and the rule handle these two traffics, Type field must be sent uncompressed.

FID	Pos	DI	TV	MO	CDF
type	1	bi		ignore	send-value

3.3. CoAP token length field

This field is bi-directional.

Several strategies can be applied to this field regarding the values:

- o no token or a wellknown length, the transmission can be avoided. A special care must be taken, if CON messages are acknowledged

with an empty ACK message. In that case the token is not always present.

FID	Pos	DI	TV	MO	CDF
TKL	1	bi	value	ignore	send-value

- o If the length is changing from one message to an other, the Token Length field must be sent. If the Token length can be limited, then only the least significant bits have to be sent. The example below allows values between 0 and 3.

FID	Pos	DI	TV	MO	CDF
TKL	1	bi	0x0	MSB(2)	LSB(2)

- o otherwise the field value has to be sent.

FID	Pos	DI	TV	MO	CDF
TKL	1	bi		ignore	value-sent

3.4. CoAP code field

This field is bidirectional, but compression can be enhanced using DI.

The CoAP Code field defines a tricky way to ensure compatibility with HTTP values. Nevertheless only 21 values are defined by [rfc7252] compared to the 255 possible values.

Code	Description	Mapping
0.00		0x00
0.01	GET	0x01
0.02	POST	0x02
0.03	PUT	0x03
0.04	DELETE	0x04
0.05	FETCH	0x05
0.06	PATCH	0x06
0.07	iPATCH	0x07
2.01	Created	0x08
2.02	Deleted	0x09
2.03	Valid	0x0A
2.04	Changed	0x0B
2.05	Content	0x0C
4.00	Bad Request	0x0D
4.01	Unauthorized	0x0E
4.02	Bad Option	0x0F
4.03	Forbidden	0x10
4.04	Not Found	0x11
4.05	Method Not Allowed	0x12
4.06	Not Acceptable	0x13
4.12	Precondition Failed	0x14
4.13	Request Entity Too Large	0x15
4.15	Unsupported Content-Format	0x16
5.00	Internal Server Error	0x17
5.01	Not Implemented	0x18
5.02	Bad Gateway	0x19
5.03	Service Unavailable	0x1A
5.04	Gateway Timeout	0x1B
5.05	Proxying Not Supported	0x1C

Figure 1: Example of CoAP code mapping

Figure 1 gives a possible mapping, it can be changed to add new codes or reduced if some values are never used by both ends. It could efficiently be coded on 5 bits.

Even if the number of code can be increase with other RFC, implementations may use a limited number of values, which can help to reduce the number of bits sent on the LPWAN.

The number of code may vary over time, some new codes may be introduced or some applications use a limited number of values.

The client and the server do not use the same values. This asymmetry can be exploited to reduce the size sent on the LPWAN.

The field can be treated differently in upstream than in downstream. If the Thing is a client an entry can be set on the uplink message with a code matching for 0.OX values and another for downlink values for Y.ZZ codes. It is the opposite if the thing is a server.

If the ES always sends or receives requests with the same method, the Code field can be elided. The entry below shows a rule for a client sending only GET request.

FID	Pos	DI	TV	MO	CDF
code	1	up	GET	equal	not-sent

If the client may send different methods, a matching-list can be applied. For table Figure 1, 3 bits are necessary, but it could be less if fewer methods are used. Example below gives an example where the ES is a server and receives only GET and POST requests.

FID	Pos	DI	TV	MO	CDF
code	1	dw	{0:0.01, 1:0.02}	match-mapping	mapping-sent

The same approach can be applied to responses.

3.5. CoAP Message ID field

This field is bidirectional.

Message ID is used for two purposes:

- o To acknowledge a CON message with an ACK.
- o To avoid duplicate messages.

In LPWAN, since a message can be received by several radio gateway, some LPWAN technologies include a sequence number in L2 to avoid duplicate frames. Therefore if the message does not need to be acknowledged (NON or RST message), the Message ID field can be avoided.

FID	Pos	DI	TV	MO	CDF
Mid	1	bi		ignore	not-sent

The decompressor must generate a value.

[[Note; check id this field is not used by OSCOAP .]]

To optimize information sent on the LPWAN, shorter values may be used during the exchange, but Message ID values generated a common CoAP implementation will not take into account this limitation. Before the compression, a proxy may be needed to reduce the size.

FID	Pos	DI	TV	MO	CDF
Mid	1	bi	0x0000	MSB(12)	LSB(4)

Otherwise if no compression is possible, the field has to be sent

FID	Pos	DI	TV	MO	CDF
Mid	1	bi		ignore	value-sent

3.6. CoAP Token field

This field is bi-directional.

Token is used to identify transactions and varies from one transaction to another. Therefore, it is usually necessary to send the value of the token field on the LPWAN network. The optimization will occur by using small values.

Common CoAP implementations may generate large tokens, even if shorter tokens could be used regarding the LPWAN characteristics. A proxy may be needed to reduce the size of the token before compression.

The size of the compress token sent is known by a combination of the Token Length field and the rule entry. For instance, with the entry below:

FID	Pos	DI	TV	MO	CDF
tkl	1	bi	2	equal	not-sent
token	1	bi	0x00	MSB(12)	LSB(4)

The uncompressed token is 2 bytes long, but the compressed size will be 4 bits.

4. CoAP options

4.1. CoAP option Content-format field.

This field is unidirectional and must not be set to bidirectional in a rule entry. It is used only by the server to inform the client about of the payload type and is never found in client requests.

If single value is expected by the client, the TV contains that value and MO is set to "equal" and the CDF is set to "not-sent". The examples below describe the rules for an ES acting as a server.

FID	Pos	DI	TV	MO	CDF
content	1	up	value	equal	not-sent

If several possible value are expected by the client, a matching-list can be used.

FID	Pos	DI	TV	MO	CDF
content	1	up	{0:50,1:41}	match-mapping	mapping-sent

Otherwise the value can be sent. The value-sent CDF in the compressor do not send the option type and the decompressor reconstruct it regarding the position in the rule.

FID	Pos	DI	TV	MO	CDF
content	1	up		ignore	value-sent

4.2. CoAP option Accept field

This field is unidirectional and must not be set to bidirectional in a rule entry. It is used only by the client to inform of the possible payload type and is never found in server response.

The number of accept options is not limited and can vary regarding the usage. To be selected a rule must contain the exact number about accept options with their positions. Since the order in which the Accept value are sent, the position order can be modified. The rule below

FID	Pos	DI	TV	MO	CDF
accept	1	up	41	egal	not-sent
accept	2	up	50	egal	not-sent

will be selected only if two accept options are in the CoAP header if this order.

The rule below:

FID	Pos	DI	TV	MO	CDF
accept	0	up	41	egal	not-sent
accept	0	up	50	egal	not-sent

will accept a-only CoAP messages with 2 accept options, but the order will not influence the rule selection. The decompression will reconstruct the header regarding the rule order.

Otherwise a matching-list can be applied to the different values, in that case the order is important to recover the appropriate value and the position must be clearly indicate.

FID	Pos	DI	TV	MO	CDF
accept 1		up	{0:50,1:41}	match-mapping	mapping-sent
accept 2		up	{0:50,1:61}	match-mapping	mapping-sent
accept 3		up	{0:61,1:71}	match-mapping	mapping-sent

Finally, the option can be explicitly sent.

FID	Pos	DI	TV	MO	CDF
accept 1	1	up		ignore	value-sent

4.3. CoAP option Max-Age field, CoAP option Uri-Host and Uri-Port fields

This field is unidirectional and must not be set to bidirectional in a rule entry. It is used only by the server to inform of the caching duration and is never found in client requests.

If the duration is known by both ends, value can be elided on the LPWAN.

A matching list can be used if some wellknown values are defined.

Otherwise the option length and value can be sent on the LPWAN.

[[note: we can reduce (or create a new option) the unit to minute, second is small for LPWAN]]

5. CoAP option Uri-Path and Uri-Query fields

This fields are unidirectional and must not be set to bidirectional in a rule entry. They are used only by the client to access to a specific resource and are never found in server response.

The Matching Operator behavior has not changed, but the value must take a position value, if the entry is repeated :

FID	Pos	DI	TV	MO	CDF
URI-Path	1	up	foo	equal	not-sent
URI-Path	2	up	bar	equal	not-sent

Figure 2: Position entry.

For instance, the rule Figure 2 matches with /foo/bar, but not /bar/foo.

When the length is not clearly indicated in the rule, the value length must be sent with the field data, which means for CoAP to send directly the CoAP option with length and value.

For instance for a CoMi path `/c/X6?k="eth0"` the rule can be set to:

FID	Pos	DI	TV	MO	CDF
URI-Path	1	up	c	equal	not-sent
URI-Path	2	up		ignore	value-sent
URI-Query	1	up	k=	MSB (16)	LSB

Figure 3: CoMi URI compression

Figure 3 shows the parsing and the compression of the URI. where c is not sent. The second element is sent with the length (i.e. 0x2 X 6) followed by the query option (i.e. 0x05 "eth0").

A Mapping list can be used to reduce size of variable Paths or Queries. In that case, to optimize the compression, several elements can be regrouped into a single entry. Numbering of elements do not change, MO comparison is set with the first element of the matching.

FID	Pos	DI	TV	MO	CDF
URI-Path	1	up	{0:"/c/c", 1:"/c/d"}	equal	not-sent
URI-Path	3	up		ignore	value-sent
URI-Query	1	up	k=	MSB (16)	LSB

Figure 4: complex path example

For instance, the following Path `/foo/bar/variable/stable` can leads to the rule defined Figure 4.

5.1. CoAP option Proxy-URI and Proxy-Scheme fields

These fields are unidirectional and must not be set to bidirectional in a rule entry. They are used only by the client to access to a specific resource and are never found in server response.

If the field value must be sent, TV is not set, MO is set to "ignore" and CDF is set to "value-sent". A mapping can also be used.

Otherwise the TV is set to the value, MO is set to "equal" and CDF is set to "not-sent"

5.2. CoAP option ETag, If-Match, If-None-Match, Location-Path and Location-Query fields

These fields are unidirectional.

These fields values cannot be stored in a rule entry. They must always be sent with the request.

[[Can include OSCOAP Object security in that category]]

6. Other RFCs

6.1. Block

Block option should be avoided in LPWAN. The minimum size of 16 bytes can be incompatible with some LPWAN technologies.

[[Note: do we recommend LPWAN fragmentation since the smallest value of 16 is too big?]]

6.2. Observe

[rfc7641] defines the Observe option. The TV is not set, MO is set to "ignore" and the CDF is set to "value-sent". SCHC does not limit the maximum size for this option (3 bytes). To reduce the transmission size either the Thing implementation should limit the value increase or a proxy can be used limit the increase.

Since RST message may be sent to inform a server that the client do not require Observe response, a rule must allow the transmission of this message.

6.3. No-Response

[rfc7967] defines an No-Response option limiting the responses made by a server to a request. If the value is not by both ends, then TV is set to this value, MO is set to "equal" and CDF is set to "not-sent".

Otherwise, if the value is changing over time, TV is not set, MO is set to "ignore" and CDF to "value-sent". A matching list can also be used to reduce the size.

7. Protocol analysis

8. Examples of CoAP header compression

8.1. Mandatory header with CON message

In this first scenario, the LPWAN compressor receives from outside client a POST message, which is immediately acknowledged by the Thing. For this simple scenario, the rules are described Figure 5.

```
rule id 1
```

Field	TV	MO	CDF	dir	Sent
CoAP version	01	equal	not-sent	bi	
CoAP Type		ignore	value-sent	bi	TT
CoAP TKL	0	equal	not-sent	bi	
CoAP Code	ML1	match-map	matching-sent	bi	CC CCC
CoAP MID	0000	MSB(7)	LSB(9)	bi	M-ID
CoAP Uri-Path	path	equal 1	not-sent	down	

Figure 5: CoAP Context to compress header without token

The version and Token Length fields are elided. Code has shrunk to 5 bits using the matching list (as the one given Figure 1: 0.01 is value 0x01 and 2.05 is value 0x0c) Message-ID has shrunk to 9 bits to preserve alignment on byte boundary. The most significant bit must be set to 0 through a CoAP proxy. Uri-Path contains a single element indicated in the matching operator.

Figure 6 shows the time diagram of the exchange. A LPWAN Application Server sends a CON message. Compression reduces the header sending only the Type, a mapped code and the least 9 significant bits of Message ID. The receiver decompresses the header. .

The CON message is a request, therefore the LC process to a dynamic mapping. When the ES receives the ACK message, this will not initiate locally a message ID mapping since it is a response. The LC receives the ACK and uncompressed it to restore the original value. Dynamic Mapping context lifetime follows the same rules as message ID duration.

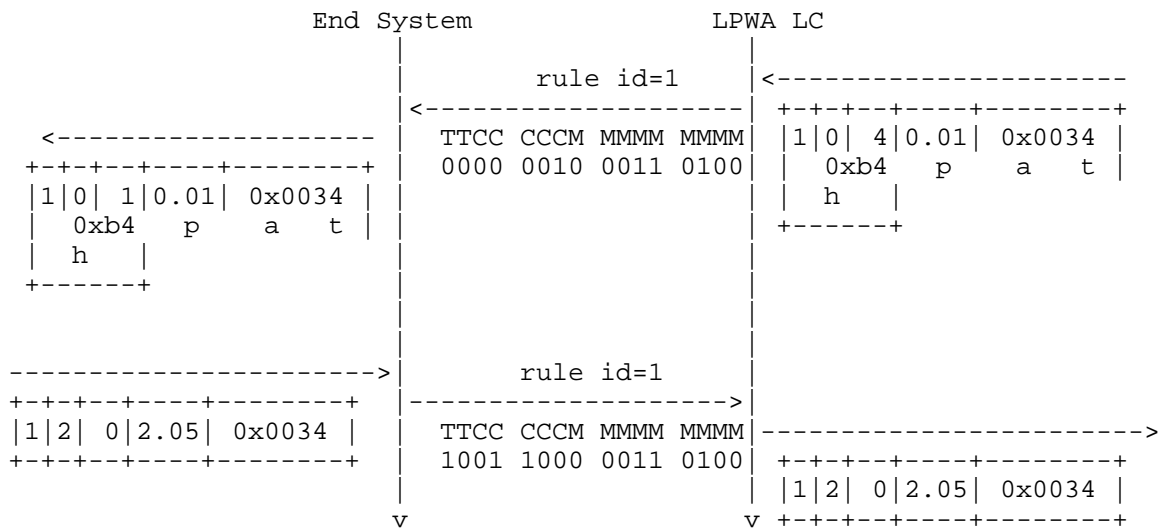


Figure 6: Compression with global addresses

The message can be further optimized by setting some fields unidirectional, as described in Figure 7. Note that Type is no more sent in the compressed format, Compressed Code size in not changed in that example (8 values are needed to code all the requests and 21 to code all the responses in the matching list Figure 1)

rule id 1

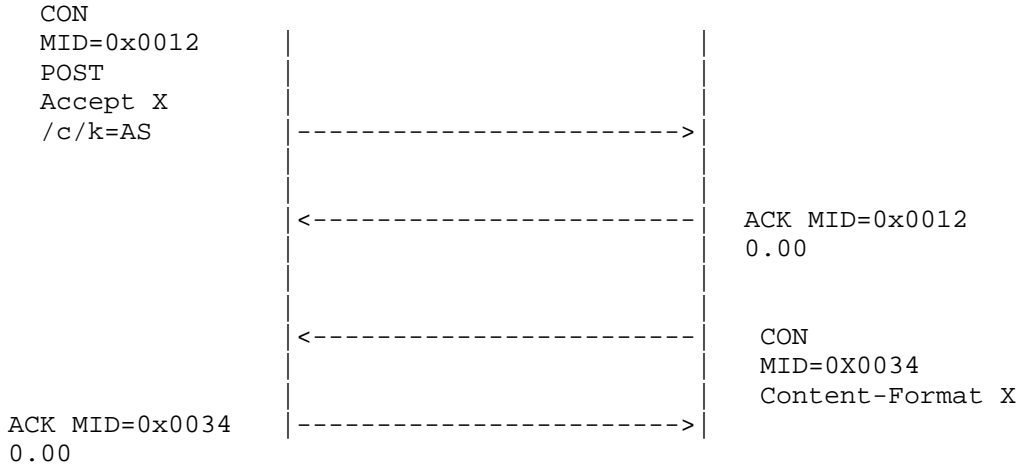
Field	TV	MO	CDF	dir	Sent
CoAP version	01	equal	not-sent	bi	
CoAP Type	CON	equal	not-sent	dw	
CoAP Type	ACK	equal	not-sent	up	
CoAP TKL	0	equal	not-sent	bi	
CoAP Code	ML2	match-map	mapping-sent	dw	CCCC C
CoAP Code	ML3	match-map	mapping-sent	up	CCCC C
CoAP MID	0000	MSB(5)	LSB(11)	bi	M-ID
CoAP Uri-Path	path	equal 1	not-sent	dw	

ML1 = {CON : 0, ACK:1} ML2 = {POST:0, 2.04:1, 0.00:3}

Figure 7: CoAP Context to compress header without token

8.2. Complete exchange

In that example, the Thing is using CoMi and sends queries for 2 SID.



rule id 3

Field	TV	MO	CDF	dir	Sent
CoAP version	01	equal	not-sent	bi	
CoAP Type	CON	equal	not-sent	up	
CoAP Type	ACK	equal	not-sent	dw	
CoAP TKL	1	equal	not-sent	bi	
CoAP Code	POST	equal	not-sent	up	
CoAP Code	0.00	equal	not-sent	dw	
CoAP MID	0000	MSB(8)	LSB	bi	MMMMMMMM
CoAP Token		ignore	send-value	up	TTTTTTTT
CoAP Uri-Path	/c	equal 1	not-sent	dw	
CoAP Uri-query	ML4	equal 1	not-sent	dw	P
CoAP Content	X	equal	not-sent	up	

rule id 4

Field	TV	MO	CDF	dir	Sent
CoAP version	01	equal	not-sent	bi	
CoAP Type	CON	equal	not-sent	dw	
CoAP Type	ACK	equal	not-sent	up	
CoAP TKL	1	equal	not-sent	bi	
CoAP Code	2.05	equal	not-sent	dw	

CoAP Code	0.00	equal	not-sent	up	
CoAP MID	0000	MSB(8)	LSB	bi	MMMMMMMM
CoAP Token		ignore	send-value	dw	TTTTTTTT
COAP Accept	X	equal	not-sent	dw	

alternative rule:

```
rule id 4
```

Field	TV	MO	CDF	dir	Sent
CoAP version	01	equal	not-sent	bi	
CoAP Type	ML1	match-map	match-sent	bi	t
CoAP TKL	1	equal	not-sent	bi	
CoAP Code	ML2	match-map	match-sent	up	cc
CoAP Code	ML3	match-map	match-sent	dw	cc
CoAP MID	0000	MSB(8)	LSB	bi	MMMMMMMM
CoAP Token		ignore	send-value	dw	TTTTTTTT
CoAP Uri-Path	/c	equal 1	not-sent	dw	
CoAP Uri-query	ML4	equal 1	not-sent	dw	P
CoAP Content	X	equal	not-sent	up	
COAP Accept	x	equal	not-sent	dw	

ML1 {CON:0, ACK:1} ML2 {POST:0, 0.00: 1} ML3 {2.05:0, 0.00:1}
 ML4 {NULL:0, k=AS:1, K=AZE:2}

9. Normative References

[I-D.toutain-lpwan-ipv6-static-context-hc]
 Minaburo, A. and L. Toutain, "LPWAN Static Context Header Compression (SCHC) for IPv6 and UDP", draft-toutain-lpwan-ipv6-static-context-hc-00 (work in progress), September 2016.

[rfc7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.

[rfc7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <<https://www.rfc-editor.org/info/rfc7641>>.

[rfc7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <<https://www.rfc-editor.org/info/rfc7967>>.

Authors' Addresses

Ana Minaburo
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

Laurent Toutain
Institut MINES TELECOM ; IMT Atlantique
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@imt-atlantique.fr

lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: June 25, 2018

A. Minaburo
Acklio
L. Toutain
IMT-Atlantique
C. Gomez
Universitat Politecnica de Catalunya
December 22, 2017

LPWAN Static Context Header Compression (SCHC) and fragmentation for
IPv6 and UDP
draft-ietf-lpwan-ipv6-static-context-hc-09

Abstract

This document describes a header compression scheme and fragmentation functionality for very low bandwidth networks. These techniques are specially tailored for Low Power Wide Area Network (LPWAN).

The Static Context Header Compression (SCHC) offers a great level of flexibility when processing the header fields. SCHC compression is based on a common static context stored in a LPWAN device and in the network. Static context means that the stored information does not change during packet transmission. The context describes the field values and keeps information that will not be transmitted through the constrained network.

SCHC must be used for LPWAN networks because it avoids complex resynchronization mechanisms, which are incompatible with LPWAN characteristics. And also, because with SCHC, in most cases IPv6/UDP headers can be reduced to a small identifier called Rule ID. Even though, sometimes, a SCHC compressed packet will not fit in one L2 PDU, and the SCHC fragmentation protocol defined in this document may be used.

This document describes the SCHC compression/decompression framework and applies it to IPv6/UDP headers. This document also specifies a fragmentation and reassembly mechanism that is used to support the IPv6 MTU requirement over LPWAN technologies. Fragmentation is mandatory for IPv6 datagrams that, after SCHC compression or when it has not been possible to apply such compression, still exceed the L2 maximum payload size. Similar solutions for other protocols such as CoAP will be described in separate documents.

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 <https://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 June 25, 2018.

Copyright Notice

Copyright (c) 2017 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 (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. LPWAN Architecture	4
3. Terminology	5
4. Static Context Header Compression	7
4.1. SCHC Rules	8
4.2. Rule ID	10
4.3. Packet processing	10
4.4. Matching operators	12
4.5. Compression Decompression Actions (CDA)	12
4.5.1. not-sent CDA	13
4.5.2. value-sent CDA	13
4.5.3. mapping-sent	14
4.5.4. LSB CDA	14
4.5.5. DEViid, APPiid CDA	14

4.5.6. Compute-*	14
5. Fragmentation	15
5.1. Overview	15
5.2. Functionalities	15
5.3. Delivery Reliability options	18
5.4. Fragmentation Frame Formats	20
5.4.1. Fragment format	20
5.4.2. ACK format	21
5.4.3. All-1 and All-0 formats	21
5.4.4. Abort formats	23
5.5. Baseline mechanism	23
5.5.1. No ACK	24
5.5.2. The Window modes	25
5.5.3. Bitmap Optimization	29
5.6. Supporting multiple window sizes	31
5.7. Downlink fragment transmission	31
6. Padding management	32
7. SCHC Compression for IPv6 and UDP headers	33
7.1. IPv6 version field	33
7.2. IPv6 Traffic class field	33
7.3. Flow label field	33
7.4. Payload Length field	34
7.5. Next Header field	34
7.6. Hop Limit field	34
7.7. IPv6 addresses fields	35
7.7.1. IPv6 source and destination prefixes	35
7.7.2. IPv6 source and destination IID	35
7.8. IPv6 extensions	36
7.9. UDP source and destination port	36
7.10. UDP length field	36
7.11. UDP Checksum field	37
8. Security considerations	37
8.1. Security considerations for header compression	37
8.2. Security considerations for fragmentation	37
9. Acknowledgements	38
10. References	38
10.1. Normative References	38
10.2. Informative References	39
Appendix A. SCHC Compression Examples	39
Appendix B. Fragmentation Examples	42
Appendix C. Fragmentation State Machines	48
Appendix D. Allocation of Rule IDs for fragmentation	55
Appendix E. Note	55
Authors' Addresses	55

1. Introduction

Header compression is mandatory to efficiently bring Internet connectivity to the node within a LPWAN network. Some LPWAN networks properties can be exploited to get an efficient header compression:

- o Topology is star-oriented; therefore, all the packets follow the same path. For the needs of this draft, the architecture can be summarized to Devices (Dev) exchanging information with LPWAN Application Server (App) through a Network Gateway (NGW).
- o Traffic flows are mostly known in advance since devices embed built-in applications. Contrary to computers or smartphones, new applications cannot be easily installed.

The Static Context Header Compression (SCHC) is defined for this environment. SCHC uses a context where header information is kept in the header format order. This context is static (the values of the header fields do not change over time) avoiding complex resynchronization mechanisms, incompatible with LPWAN characteristics. In most of the cases, IPv6/UDP headers are reduced to a small context identifier.

The SCHC header compression mechanism is independent of the specific LPWAN technology over which it will be used.

LPWAN technologies are also characterized, among others, by a very reduced data unit and/or payload size [I-D.ietf-lpwan-overview]. However, some of these technologies do not support layer two fragmentation, therefore the only option for them to support the IPv6 MTU requirement of 1280 bytes [RFC2460] is the use of a fragmentation protocol at the adaptation layer below IPv6. This draft defines also a fragmentation functionality to support the IPv6 MTU requirement over LPWAN technologies. Such functionality has been designed under the assumption that data unit reordering will not happen between the entity performing fragmentation and the entity performing reassembly.

2. LPWAN Architecture

LPWAN technologies have similar architectures but different terminology. We can identify different types of entities in a typical LPWAN network, see Figure 1:

- o Devices (Dev) are the end-devices or hosts (e.g. sensors, actuators, etc.). There can be a high density of devices per radio gateway.

- o The Radio Gateway (RGW), which is the end point of the constrained link.
- o The Network Gateway (NGW) is the interconnection node between the Radio Gateway and the Internet.
- o LPWAN-AAA Server, which controls the user authentication and the applications.
- o Application Server (App)

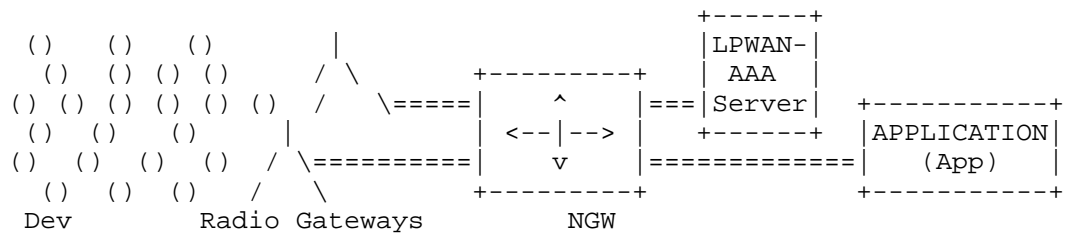


Figure 1: LPWAN Architecture

3. Terminology

This section defines the terminology and acronyms used in this document.

- o All-0. Fragment format for the last frame of a window.
- o All-1. Fragment format for the last frame of a packet.
- o All-0 empty. Fragment format without payload for requesting the Bitmap when the Retransmission Timer expires in a window that is not the last one for a fragmented packet transmission.
- o All-1 empty. Fragment format without payload for requesting the Bitmap when the Retransmission Timer expires in the last window.
- o App: LPWAN Application. An application sending/receiving IPv6 packets to/from the Device.
- o APP-IID: Application Interface Identifier. Second part of the IPv6 address to identify the application interface
- o Bi: Bidirectional, a rule entry that applies in both directions.

- o C: Checked bit. Used in an acknowledgment (ACK) header to determine when the MIC is correct (1) or not (0).
- o CDA: Compression/Decompression Action. An action that is performed for both functionalities to compress a header field or to recover its original value in the decompression phase.
- o Context: A set of rules used to compress/decompress headers
- o Dev: Device. A Node connected to the LPWAN. A Dev may implement SCHC.
- o Dev-IID: Device Interface Identifier. Second part of the IPv6 address to identify the device interface
- o DI: Direction Indicator is a differentiator for matching in order to be able to have different values for both sides.
- o DTag: Datagram Tag is a fragmentation header field that is set to the same value for all fragments carrying the same IPv6 datagram.
- o Dw: Down Link direction for compression, from SCHC C/D to Dev
- o FCN: Fragment Compressed Number is a fragmentation header field that carries an efficient representation of a larger-sized fragment number.
- o FID: Field Identifier is an index to describe the header fields in the Rule
- o FL: Field Length is a value to identify if the field is fixed or variable length.
- o FP: Field Position is a value that is used to identify each instance a field appears in the header.
- o IID: Interface Identifier. See the IPv6 addressing architecture [RFC7136]
- o Inactivity Timer. A timer to end the fragmentation state machine when there is an error and there is no possibility to continue an on-going fragmented packet transmission.
- o MIC: Message Integrity Check. A fragmentation header field computed over an IPv6 packet before fragmentation, used for error detection after IPv6 packet reassembly.

- o MO: Matching Operator. An operator used to match a value contained in a header field with a value contained in a Rule.
- o Retransmission Timer. A timer used by the fragment sender during an on-going fragmented packet transmission to detect possible link errors when waiting for a possible incoming ACK.
- o Rule: A set of header field values.
- o Rule entry: A row in the rule that describes a header field.
- o Rule ID: An identifier for a rule, SCHC C/D, and Dev share the same Rule ID for a specific flow. A set of Rule IDs are used to support fragmentation functionality.
- o SCHC C/D: Static Context Header Compression Compressor/Decompressor. A process in the network to achieve compression/decompressing headers. SCHC C/D uses SCHC rules to perform compression and decompression.
- o TV: Target value. A value contained in the Rule that will be matched with the value of a header field.
- o Up: Up Link direction for compression, from Dev to SCHC C/D.
- o W: Window bit. A fragment header field used in Window mode (see section 5), which carries the same value for all fragments of a window.
- o Window: A subset of the fragments needed to carry a packet (see section 5)

4. Static Context Header Compression

Static Context Header Compression (SCHC) avoids context synchronization, which is the most bandwidth-consuming operation in other header compression mechanisms such as RoHC [RFC5795]. Based on the fact that the nature of data flows is highly predictable in LPWAN networks, some static contexts may be stored on the Device (Dev). The contexts must be stored in both ends, and it can either be learned by a provisioning protocol or by out of band means or it can be pre-provisioned, etc. The way the context is learned on both sides are out of the scope of this document.

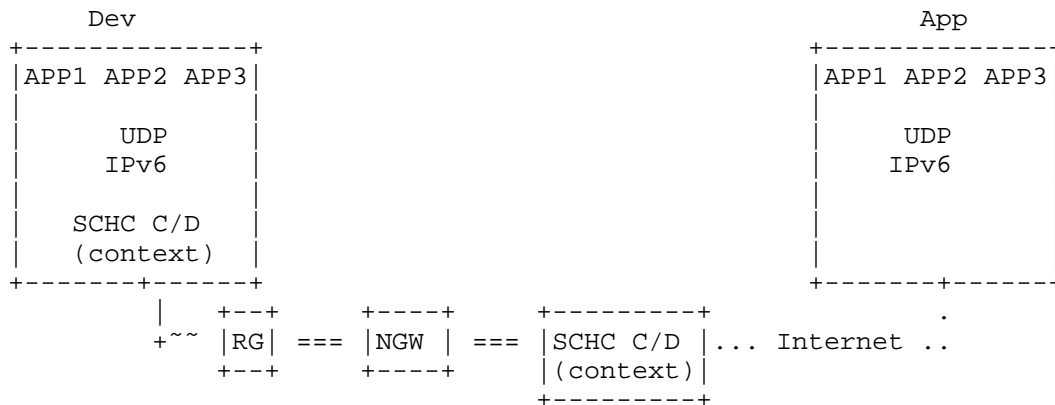


Figure 2: Architecture

Figure 2 represents the architecture for compression/decompression, it is based on [I-D.ietf-lpwan-overview] terminology. The Device is sending applications flows using IPv6 or IPv6/UDP protocols. These flows are compressed by a Static Context Header Compression Compressor/Decompressor (SCHC C/D) to reduce headers size. The resulting information is sent to a layer two (L2) frame to a LPWAN Radio Network (RG) which forwards the frame to a Network Gateway (NGW). The NGW sends the data to an SCHC C/D for decompression which shares the same rules with the Dev. The SCHC C/D can be located on the Network Gateway (NGW) or in another place as long as a tunnel is established between the NGW and the SCHC C/D. The SCHC C/D in both sides must share the same set of Rules. After decompression, the packet can be sent on the Internet to one or several LPWAN Application Servers (App).

The SCHC C/D process is bidirectional, so the same principles can be applied in the other direction.

4.1. SCHC Rules

The main idea of the SCHC compression scheme is to send the Rule id to the other end instead of sending known field values. This Rule id identifies a rule that matches as much as possible the original packet values. When a value is known by both ends, it is not necessary to send it through the LPWAN network.

The context contains a list of rules (cf. Figure 3). Each Rule contains itself a list of fields descriptions composed of a field identifier (FID), a field length (FL), a field position (FP), a direction indicator (DI), a target value (TV), a matching operator (MO) and a Compression/Decompression Action (CDA).

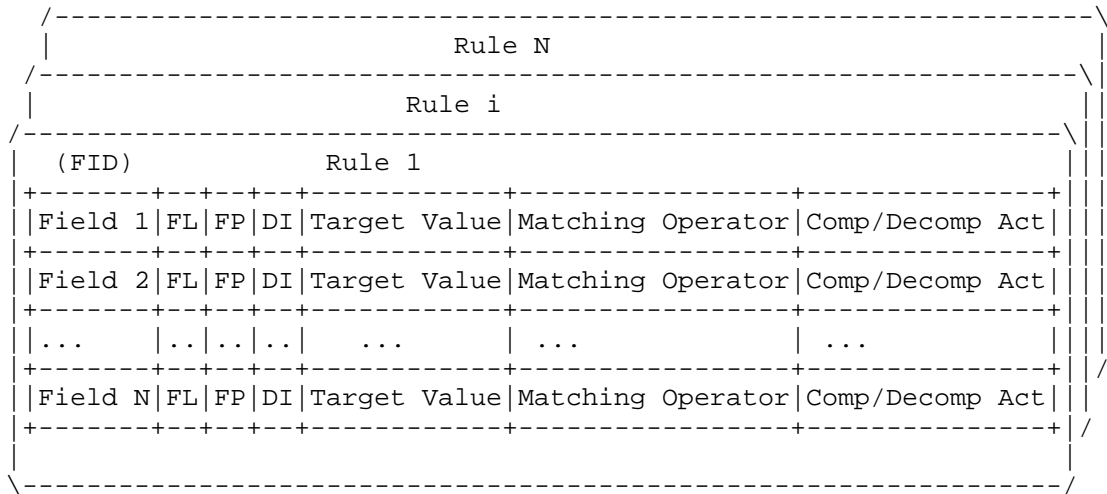


Figure 3: Compression/Decompression Context

The Rule does not describe the original packet format which must be known from the compressor/decompressor. The rule just describes the compression/decompression behavior for the header fields. In the rule, the description of the header field should be performed in the format packet order.

The Rule also describes the compressed header fields which are transmitted regarding their position in the rule which is used for data serialization on the compressor side and data deserialization on the decompressor side.

The Context describes the header fields and its values with the following entries:

- o A Field ID (FID) is a unique value to define the header field.
- o A Field Length (FL) is the length of the field that can be of fixed length as in IPv6 or UDP headers or variable length as in CoAP options. Fixed length fields shall be represented by its actual value in bits. Variable length fields shall be represented by a function or a variable.
- o A Field Position (FP) indicating if several instances of the field exist in the headers which one is targeted. The default position is 1
- o A direction indicator (DI) indicating the packet direction. Three values are possible:

- * UPLINK (Up) when the field or the value is only present in packets sent by the Dev to the App,
 - * DOWNLINK (Dw) when the field or the value is only present in packet sent from the App to the Dev and
 - * BIDIRECTIONAL (Bi) when the field or the value is present either upstream or downstream.
- o A Target Value (TV) is the value used to make the comparison with the packet header field. The Target Value can be of any type (integer, strings, etc.). For instance, it can be a single value or a more complex structure (array, list, etc.), such as a JSON or a CBOR structure.
 - o A Matching Operator (MO) is the operator used to make the comparison between the Field Value and the Target Value. The Matching Operator may require some parameters. MO is only used during the compression phase.
 - o A Compression Decompression Action (CDA) is used to describe the compression and the decompression process. The CDA may require some parameters, CDA are used in both compression and decompression phases.

4.2. Rule ID

Rule IDs are sent between both compression/decompression elements. The size of the Rule ID is not specified in this document, it is implementation-specific and can vary regarding the LPWAN technology, the number of flows, among others.

Some values in the Rule ID space are reserved for other functionalities than header compression as fragmentation. (See Section 5).

Rule IDs are specific to a Dev. Two Devs may use the same Rule ID for different header compression. To identify the correct Rule ID, the SCHC C/D needs to combine the Rule ID with the Dev L2 identifier to find the appropriate Rule.

4.3. Packet processing

The compression/decompression process follows several steps:

- o compression Rule selection: The goal is to identify which Rule(s) will be used to compress the packet's headers. When doing compression in the NGW side the SCHC C/D needs to find the correct

Rule to be used by identifying its Dev-ID and the Rule-ID. In the Dev, only the Rule-ID may be used. The next step is to choose the fields by their direction, using the direction indicator (DI), so the fields that do not correspond to the appropriated DI will be excluded. Next, then the fields are identified according to their field identifier (FID) and field position (FP). If the field position does not correspond, then the Rule is not used and the SCHC take next Rule. Once the DI and the FP correspond to the header information, each field's value is then compared to the corresponding target value (TV) stored in the Rule for that specific field using the matching operator (MO). If all the fields in the packet's header satisfy all the matching operators (MOs) of a Rule (i.e. all results are True), the fields of the header are then processed according to the Compression/Decompression Actions (CDAs) and a compressed header is obtained. Otherwise, the next rule is tested. If no eligible rule is found, then the header must be sent without compression, in which case the fragmentation process must be required.

- o sending: The Rule ID is sent to the other end followed by the information resulting from the compression of header fields, directly followed by the payload. The product of field compression is sent in the order expressed in the Rule for the matching fields. The way the Rule ID is sent depends on the specific LPWAN layer two technology and will be specified in a specific document and is out of the scope of this document. For example, it can be either included in a Layer 2 header or sent in the first byte of the L2 payload. (Cf. Figure 4).
- o decompression: In both directions, the receiver identifies the sender through its device-id (e.g. MAC address) and selects the appropriate Rule through the Rule ID. This Rule gives the compressed header format and associates these values to the header fields. It applies the CDA action to reconstruct the original header fields. The CDA application order can be different from the order given by the Rule. For instance, Compute-* may be applied at the end, after all the other CDAs.

If after using SCHC compression and adding the payload to the L2 frame the datagram is not multiple of 8 bits, padding may be used.

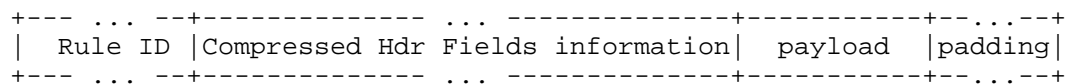


Figure 4: LPWAN Compressed Format Packet

4.4. Matching operators

Matching Operators (MOs) are functions used by both SCHC C/D endpoints involved in the header compression/decompression. They are not typed and can be applied indifferently to integer, string or any other data type. The result of the operation can either be True or False. MOs are defined as follows:

- o equal: A field value in a packet matches with a TV in a Rule if they are equal.
- o ignore: No check is done between a field value in a packet and a TV in the Rule. The result of the matching is always true.
- o MSB(length): A matching is obtained if the most significant bits of the length field value bits of the header are equal to the TV in the rule. The MSB Matching Operator needs a parameter, indicating the number of bits, to proceed to the matching.
- o match-mapping: The goal of mapping-sent is to reduce the size of a field by allocating a shorter value. The Target Value contains a list of values. Each value is identified by a short ID (or index). This operator matches if a field value is equal to one of those target values.

4.5. Compression Decompression Actions (CDA)

The Compression Decompression Action (CDA) describes the actions taken during the compression of headers fields, and inversely, the action taken by the decompressor to restore the original value.

Action	Compression	Decompression
not-sent	elided	use value stored in ctxt
value-sent	send	build from received value
mapping-sent	send index	value from index on a table
LSB(length)	send LSB	TV OR received value
compute-length	elided	compute length
compute-checksum	elided	compute UDP checksum
Deviid	elided	build IID from L2 Dev addr
Appiid	elided	build IID from L2 App addr

Figure 5: Compression and Decompression Functions

Figure 5 summarizes the basics functions defined to compress and decompress a field. The first column gives the action's name. The second and third columns outline the compression/decompression behavior.

Compression is done in the rule order and compressed values are sent in that order in the compressed message. The receiver must be able to find the size of each compressed field which can be given by the rule or may be sent with the compressed header.

If the field is identified as being variable, then its size must be sent first using the following coding:

- o If the size is between 0 and 14 bytes it is sent using 4 bits.
- o For values between 15 and 255, the first 4 bits sent are set to 1 and the size is sent using 8 bits.
- o For higher value, the first 12 bits are set to 1 and the size is sent on 2 bytes.

4.5.1. not-sent CDA

The not-sent function is generally used when the field value is specified in the rule and therefore known by the both Compressor and Decompressor. This action is generally used with the "equal" MO. If MO is "ignore", there is a risk to have a decompressed field value different from the compressed field.

The compressor does not send any value in the compressed header for the field on which compression is applied.

The decompressor restores the field value with the target value stored in the matched rule.

4.5.2. value-sent CDA

The value-sent action is generally used when the field value is not known by both Compressor and Decompressor. The value is sent in the compressed message header. Both Compressor and Decompressor must know the size of the field, either implicitly (the size is known by both sides) or explicitly in the compressed header field by indicating the length. This function is generally used with the "ignore" MO.

4.5.3. mapping-sent

The mapping-sent is used to send a smaller index associated with the list of values in the Target Value. This function is used together with the "match-mapping" MO.

The compressor looks on the TV to find the field value and send the corresponding index. The decompressor uses this index to restore the field value.

The number of bits sent is the minimal size for coding all the possible indexes.

4.5.4. LSB CDA

LSB action is used to avoid sending the known part of the packet field header to the other end. This action is used together with the "MSB" MO. A length can be specified in the rule to indicate how many bits have to be sent. If the length is not specified, the number of bits sent is the field length minus the bits' length specified in the MSB MO.

The compressor sends the "length" Least Significant Bits. The decompressor combines the value received with the Target Value.

If this action is made on a variable length field, the remaining size in byte has to be sent before.

4.5.5. DEViid, APPiid CDA

These functions are used to process respectively the Dev and the App Interface Identifiers (Deviid and Appiid) of the IPv6 addresses. Appiid CDA is less common since current LPWAN technologies frames contain a single address.

The IID value may be computed from the Device ID present in the Layer 2 header. The computation is specific for each LPWAN technology and may depend on the Device ID size.

In the downstream direction, these CDA may be used to determine the L2 addresses used by the LPWAN.

4.5.6. Compute-*

These classes of functions are used by the decompressor to compute the compressed field value based on received information. Compressed fields are elided during compression and reconstructed during decompression.

- o compute-length: compute the length assigned to this field. For instance, regarding the field ID, this CDA may be used to compute IPv6 length or UDP length.
- o compute-checksum: compute a checksum from the information already received by the SCHC C/D. This field may be used to compute UDP checksum.

5. Fragmentation

5.1. Overview

In LPWAN technologies, the L2 data unit size typically varies from tens to hundreds of bytes. If after applying SCHC header compression or when SCHC header compression is not possible the entire IPv6 datagram fits within a single L2 data unit, the fragmentation mechanism is not used and the packet is sent. Otherwise, the datagram SHALL be broken into fragments.

LPWAN technologies impose some strict limitations on traffic, (e.g.) devices are sleeping most of the time and may receive data during a short period of time after transmission to preserve battery. To adapt the SCHC fragmentation to the capabilities of LPWAN technologies, it is desirable to enable optional fragment retransmission and to allow a gradation of fragment delivery reliability. This document does not make any decision with regard to which fragment delivery reliability option(s) will be used over a specific LPWAN technology.

An important consideration is that LPWAN networks typically follow a the star topology, and therefore data unit reordering is not expected in such networks. This specification assumes that reordering will not happen between the entity performing fragmentation and the entity performing reassembly. This assumption allows to reduce complexity and overhead of the fragmentation mechanism.

5.2. Functionalities

This subsection describes the different fields in the fragmentation header frames (see the related formats in Section 5.4), as well as the tools that are used to enable the fragmentation functionalities defined in this document, and the different reliability options supported.

- o Rule ID. The Rule ID is present in the fragment header and in the ACK header format. The Rule ID in a fragment header is used to identify that a fragment is being carried, the fragmentation delivery reliability option used and it may indicate the window

size in use (if any). The Rule ID in the fragmentation header also allows to interleave non-fragmented IPv6 datagrams with fragments that carry a larger IPv6 datagram. The Rule ID in an ACK allows to identify that the message is an ACK.

- o Fragment Compressed Number (FCN). The FCN is included in all fragments. This field can be understood as a truncated, efficient representation of a larger-sized fragment number, and does not carry an absolute fragment number. There are two FCN reserved values that are used for controlling the fragmentation process, as described next. The FCN value with all the bits equal to 1 (All-1) denotes the last fragment of a packet. And the FCN value with all the bits equal to 0 (All-0) denotes the last fragment of a window (when such window is not the last one of the packet) in any window mode or the fragments in No ACK mode. The rest of the FCN values are assigned in a sequential and decreasing order, which has the purpose to avoid possible ambiguity for the receiver that might arise under certain conditions. In the fragments, this field is an unsigned integer, with a size of N bits. In the No ACK mode it is set to 1 bit (N=1). For the other reliability options, it is recommended to use a number of bits (N) equal to or greater than 3. Nevertheless, the appropriate value will be defined in the corresponding technology documents. The FCN MUST be set sequentially decreasing from the highest FCN in the window (which will be used for the first fragment), and MUST wrap from 0 back to the highest FCN in the window.
For windows that are not the last one from a fragmented packet, the FCN for the last fragment in such windows is an All-0. This indicates that the window is finished and communication proceeds according to the reliability option in use. The FCN for the last fragment in the last window is an All-1. It is also important to note that, for No ACK mode or N=1, the last fragment of the packet will carry a FCN equal to 1, while all previous fragments will carry a FCN of 0.
- o Datagram Tag (DTag). The DTag field, if present, is set to the same value for all fragments carrying the same IPv6 datagram. This field allows to interleave fragments that correspond to different IPv6 datagrams. In the fragment formats the size of the DTag field is T bits, which may be set to a value greater than or equal to 0 bits. DTag MUST be set sequentially increasing from 0 to $2^T - 1$, and MUST wrap back from $2^T - 1$ to 0. In the ACK format, DTag carries the same value as the DTag field in the fragments for which this ACK is intended.
- o W (window): W is a 1-bit field. This field carries the same value for all fragments of a window, and it is complemented for the next window. The initial value for this field is 0. In the ACK

format, this field also has a size of 1 bit. In all ACKs, the W bit carries the same value as the W bit carried by the fragments whose reception is being positively or negatively acknowledged by the ACK.

- o Message Integrity Check (MIC). This field, which has a size of M bits, is computed by the sender over the complete packet (i.e. a SCHC compressed or an uncompressed IPv6 packet) before fragmentation. The MIC allows the receiver to check errors in the reassembled packet, while it also enables compressing the UDP checksum by use of SCHC compression. The CRC32 as 0xEDB88320 is recommended as the default algorithm for computing the MIC. Nevertheless, other algorithm MAY be mandated in the corresponding technology documents (e.g. technology-specific profiles).
- o C (MIC checked): C is a 1-bit field. This field is used in the ACK format packets to report the outcome of the MIC check, i.e. whether the reassembled packet was correctly received or not.
- o Retransmission Timer. It is used by a fragment sender after the transmission of a window to detect a transmission error of the ACK corresponding to this window. Depending on the reliability option, it will lead to a request for an ACK retransmission on ACK-Always or it will trigger the next window on ACK-on-error. The duration of this timer is not defined in this document and must be defined in the corresponding technology documents (e.g. technology-specific profiles).
- o Inactivity Timer. This timer is used by a fragment receiver to detect when there is a problem in the transmission of fragments and the receiver does not get any fragment during a period of time or a number of packets in a period of time. When this happens, an Abort message needs to be sent. Initially, and each time a fragment is received the timer is reinitialized. The duration of this timer is not defined in this document and must be defined in the specific technology document (e.g. technology-specific profiles).
- o Attempts. It is a counter used to request a missing ACK, and in consequence to determine when an Abort is needed, because there are recurrent fragment transmission errors, whose maximum value is MAX_ACK_REQUESTS. The default value of MAX_ACK_REQUESTS is not stated in this document, and it is expected to be defined in other documents (e.g. technology-specific profiles). The Attempts counter is defined per window, it will be initialized each time a new window is used.

- o Bitmap. The Bitmap is a sequence of bits carried in an ACK for a given window. Each bit in the Bitmap corresponds to a fragment of the current window, and provides feedback on whether the fragment has been received or not. The right-most position on the Bitmap is used to report whether the All-0 or All-1 fragments have been received or not. Feedback for a fragment with the highest FCN value is provided by the left-most position in the Bitmap. In the Bitmap, a bit set to 1 indicates that the corresponding FCN fragment has been correctly sent and received. However, the sending format of the Bitmap will be truncated until a byte boundary where the last error is given. However, when all the Bitmap is transmitted, it may be truncated, see more details in Section 5.5.3
- o Abort. In case of error or when the Inactivity timer expires or MAX_ACK_REQUESTS is reached the sender or the receiver may use the Abort frames. When the receiver needs to abort the on-going fragmented packet transmission, it uses the ACK Abort format packet with all the bits set to 1. When the sender needs to abort the transmission it will use the All-1 Abort format, this fragment is not acked.
- o Padding (P). Padding will be used to align the last byte of a fragment with a byte boundary. The number of bits used for padding is not defined and depends on the size of the Rule ID, DTag and FCN fields, and on the layer two payload size.

5.3. Delivery Reliability options

This specification defines the following three fragment delivery reliability options:

- o No ACK. No ACK is the simplest fragment delivery reliability option. The receiver does not generate overhead in the form of acknowledgments (ACKs). However, this option does not enhance delivery reliability beyond that offered by the underlying LPWAN technology. In the No ACK option, the receiver MUST NOT issue ACKs.
- o Window mode - ACK always (ACK-Always). The ACK-always option provides flow control. In addition, this option is able to handle long bursts of lost fragments, since detection of such events can be done before the end of the IPv6 packet transmission, as long as the window size is short enough. However, such benefit comes at the expense of ACK use. In ACK-always, an ACK is transmitted by the fragment receiver after a window of fragments has been sent. A window of fragments is a subset of the full set of fragments needed to carry an IPv6

packet. In this mode, the ACK informs the sender about received and/or missed fragments from the window of fragments. Upon receipt of an ACK that informs about any lost fragments, the sender retransmits the lost fragments. When an ACK is not received by the fragment sender, the latter sends an ACK request using the All-1 empty fragment.

The maximum number of ACK requests is MAX_ACK_REQUESTS.

- o Window mode - ACK-on-error (ACK-on-error). The ACK-on-error option is suitable for links offering relatively low L2 data unit loss probability. This option reduces the number of ACKs transmitted by the fragment receiver. This may be especially beneficial in asymmetric scenarios, e.g. where fragmented data are sent uplink and the underlying LPWAN technology downlink capacity or message rate is lower than the uplink one. In ACK-on-error, an ACK is transmitted by the fragment receiver after a window of fragments have been sent, only if at least one of the fragments in the window has been lost. In this mode, the ACK informs the sender about received and/or missed fragments from the window of fragments. Upon receipt of an ACK that informs about any lost fragments, the sender retransmits the lost fragments. However, if an ACK is lost, the sender assumes that all fragments covered by the ACK have been successfully delivered, and the receiver will abort the on-going fragmented packet transmission. One exception to this behavior is in the last window, where the receiver MUST transmit an ACK, even if all the fragments in the last window have been correctly received.

The same reliability option MUST be used for all fragments of a packet. It is up to implementers and/or representatives of the underlying LPWAN technology to decide which reliability option to use and whether the same reliability option applies to all IPv6 packets or not. Note that the reliability option to be used is not necessarily tied to the particular characteristics of the underlying L2 LPWAN technology (e.g. the No ACK reliability option may be used on top of an L2 LPWAN technology with symmetric characteristics for uplink and downlink).

This document does not make any decision as to which fragment delivery reliability option(s) are supported by a specific LPWAN technology.

Examples of the different reliability options described are provided in Appendix B.

5.4. Fragmentation Frame Formats

This section defines the fragment format, the All-0 and All-1 frame formats, the ACK format and the Abort frame formats.

5.4.1. Fragment format

A fragment comprises a fragment header, a fragment payload, and Padding bits (if any). A fragment conforms to the format shown in Figure 6. The fragment payload carries a subset of either a SCHC header or an IPv6 header or the original IPv6 packet data payload. A fragment is the payload in the L2 protocol data unit (PDU).

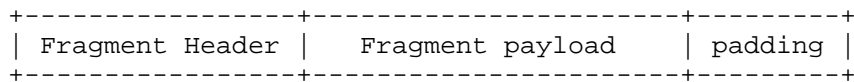


Figure 6: Fragment format.

In the No ACK option, fragments except the last one SHALL contain the format as defined in Figure 7. The total size of the fragment header is R bits.

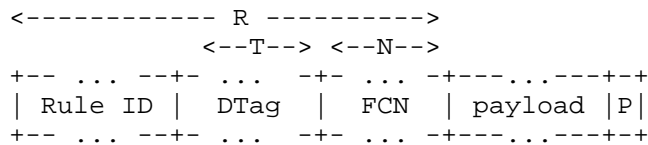


Figure 7: Fragment Format for Fragments except the Last One, No ACK option

In any of the Window mode options, fragments except the last one SHALL contain the fragmentation format as defined in Figure 8. The total size of the fragment header in this format is R bits. .

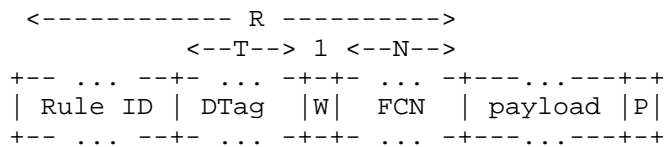


Figure 8: Fragment Format for Fragments except the Last One, Window mode

5.4.2. ACK format

The format of an ACK that acknowledges a window that is not the last one (denoted as All-0 window) is shown in Figure 9.

```

<----- R ----->
      <- T -> 1
+---- ... ---+... -++----- ... ----+
| Rule ID | DTag |W|   Bitmap   | (no payload)
+---- ... ---+... -++----- ... ----+

```

Figure 9: ACK format for All-0 windows

To acknowledge the last window of a packet (denoted as All-1 window), a C bit (i.e. MIC checked) following the W bit is set to 1 to indicate that the MIC check computed by the receiver matches the MIC present in the All-1 fragment. If the MIC check fails, the C bit is set to 0 and the Bitmap for the All-1 window follows.

```

<----- R -----> <- byte boundary ->
      <- T -> 1 1
+---- ... ---+... -++---+
| Rule ID | DTag |W|1| (MIC correct)
+---- ... ---+... -++---+

+---- ... ---+... -++---+----- ... -----+
| Rule ID | DTag |W|0|   Bitmap   | (MIC Incorrect)
+---- ... ---+... -++---+----- ... -----+
                                C

```

Figure 10: Format of an ACK for All-1 windows

5.4.3. All-1 and All-0 formats

The All-0 format is used for the last fragment of a window that is not the last window of the packet.

```

<----- R ----->
      <- T -> 1 <- N ->
+-- ... ---+ ... -++--- ... -+---- ... ----+
| Rule ID | DTag |W| 0..0 | payload |
+-- ... ---+ ... -++--- ... -+---- ... ----+

```

Figure 11: All-0 fragment format

The All-0 empty fragment format is used by a sender to request an ACK in ACK-Always mode

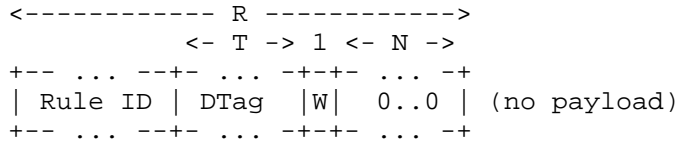


Figure 12: All-0 empty fragment format

In the No ACK option, the last fragment of an IPv6 datagram SHALL contain a fragment header that conforms to the format shown in Figure 13. The total size of this fragment header is R+M bits.

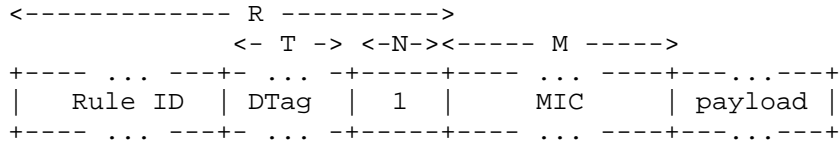


Figure 13: All-1 Fragment Format for the Last Fragment, No ACK option

In any of the Window modes, the last fragment of an IPv6 datagram SHALL contain a fragment header that conforms to the format shown in Figure 14. The total size of the fragment header in this format is R+M bits.

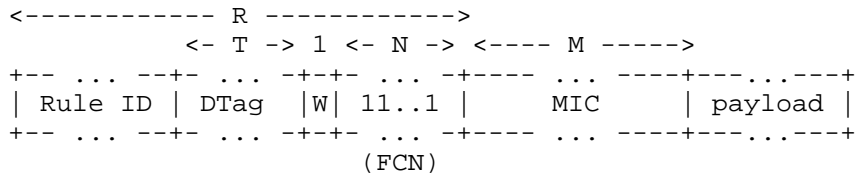


Figure 14: All-1 Fragment Format for the Last Fragment, Window mode

In either ACK-Always or ACK-on-error, in order to request a retransmission of the ACK for the All-1 window, the fragment sender uses the format shown in Figure 15. The total size of the fragment header in this format is R+M bits.

```

<----- R ----->
      <- T -> 1 <- N -> <---- M ---->
+--- ... ---+ ... ---+ ... ---+ ... ---+
| Rule ID | DTag |W| 1..1 |      MIC      | (no payload)
+--- ... ---+ ... ---+ ... ---+ ... ---+
    
```

Figure 15: All-1 for Retries format, also called All-1 empty

The values for R, N, T and M are not specified in this document, and have to be determined in other documents (e.g. technology-specific profile documents).

5.4.4. Abort formats

The All-1 Abort and the ACK abort messages have the following formats.

```

<----- byte boundary -----><---- 1 byte --->
+---- ... ---+ ... ---+...---+...---+...---+...---+
| Rule ID | DTag |W| FCN |      FF      | (no MIC & no payload)
+---- ... ---+ ... ---+...---+...---+...---+...---+
    
```

Figure 16: All-1 Abort format

```

<----- byte boundary -----><---- 1 byte --->
+---- ... ---+... ---+...---+...---+...---+...---+
| Rule ID | DTag |W| 1..1|      FF      |
+---- ... ---+... ---+...---+...---+...---+...---+
    
```

Figure 17: ACK Abort format

5.5. Baseline mechanism

The fragment receiver needs to identify all the fragments that belong to a given IPv6 datagram. To this end, the receiver SHALL use:

- o The sender’s L2 source address (if present),
- o The destination’s L2 address (if present),
- o Rule ID and
- o DTag (the latter, if present).

Then, the fragment receiver may determine the fragment delivery reliability option that is used for this fragment based on the Rule ID field in that fragment.

Upon receipt of a link fragment, the receiver starts constructing the original unfragmented packet. It uses the FCN and the order of arrival of each fragment to determine the location of the individual fragments within the original unfragmented packet. A fragment payload may carry bytes from a SCHC compressed IPv6 header, an uncompressed IPv6 header or an IPv6 datagram data payload. An unfragmented packet could be a SCHC compressed or an uncompressed IPv6 packet (header and data). For example, the receiver may place the fragment payload within a payload datagram reassembly buffer at the location determined from: the FCN, the arrival order of the fragments, and the fragment payload sizes. In Window mode, the fragment receiver also uses the W bit in the received fragments. Note that the size of the original, unfragmented packet cannot be determined from fragmentation headers.

Fragmentation functionality uses the FCN value, which has a length of N bits. The All-1 and All-0 FCN values are used to control the fragmentation transmission. The FCN will be assigned sequentially in a decreasing order starting from $2^N - 2$, i.e. the highest possible FCN value depending on the FCN number of bits, but excluding the All-1 value. In all modes, the last fragment of a packet must contain a MIC which is used to check if there are errors or missing fragments, and must use the corresponding All-1 fragment format. Also note that, a fragment with an All-0 format is considered the last fragment of the current window.

If the recipient receives the last fragment of a datagram (All-1), it checks for the integrity of the reassembled datagram, based on the MIC received. In No ACK, if the integrity check indicates that the reassembled datagram does not match the original datagram (prior to fragmentation), the reassembled datagram MUST be discarded. In Window mode, a MIC check is also performed by the fragment receiver after reception of each subsequent fragment retransmitted after the first MIC check.

5.5.1. No ACK

In the No ACK mode there is no feedback communication from the fragment receiver. The sender will send the fragments of a packet until the last one without any possibility to know if errors or a losses have occurred. As in this mode there is not a need to identify specific fragments a one-bit FCN is used, therefore FCN All-0 will be used in all fragments except the last one. The latter will carry an All-1 FCN and will also carry the MIC. The receiver

will wait for fragments and will set the Inactivity timer. The No ACK mode will use the MIC contained in the last fragment to check error. When the Inactivity Timer expires or when the MIC check indicates that the reassembled packet does not match the original one, the receiver will release all resources allocated to reassembly of the packet. The initial value of the Inactivity Timer will be determined based on the characteristics of the underlying LPWAN technology and will be defined in other documents (e.g. technology-specific profile documents).

5.5.2. The Window modes

In Window modes, a jumping window protocol uses two windows alternatively, identified as 0 and 1. A fragment with all FCN bits set to 0 (i.e. an All-0 fragment) indicates that the window is over (i.e. the fragment is the last one of the window) and allows to switch from one window to the next one. The All-1 FCN in a fragment indicates that it is the last fragment of the packet being transmitted and therefore there will not be another window for the packet.

The Window mode offers two different reliability option modes: ACK-on-error and ACK-always.

5.5.2.1. ACK-Always

In ACK-Always, the sender sends fragments by using the two-jumping window procedure. A delay between each fragment can be added to respect regulation rules or constraints imposed by the applications. Each time a fragment is sent, the FCN is decreased by one. When the FCN reaches value 0 and there are more fragments to be sent, an All-0 fragment is sent and the Retransmission Timer is set. The sender waits for an ACK to know if transmission errors have occurred. Then, the receiver sends an ACK reporting whether any fragments have been lost or not by setting the corresponding bits in the Bitmap, otherwise, an ACK without Bitmap will be sent, allowing transmission of a new window. When the last fragment of the packet is sent, an All-1 fragment (which includes a MIC) is used. In that case, the sender sets the Retransmission Timer to wait for the ACK corresponding to the last window. During this period, the sender starts listening to the radio and starts the Retransmission Timer, which needs to be dimensioned based on the received window available for the LPWAN technology in use. If the Retransmission Timer expires, an empty All-0 (or an empty All-1 if the last fragment has been sent) fragment is sent to ask the receiver to resend its ACK. The window number is not changed.

When the sender receives an ACK, it checks the W bit carried by the ACK. Any ACK carrying an unexpected W bit is discarded. If the W bit value of the received ACK is correct, the sender analyzes the received Bitmap. If all the fragments sent during the window have been well received, and if at least one more fragment needs to be sent, the sender moves its sending window to the next window value and sends the next fragments. If no more fragments have to be sent, then the fragmented packet transmission is finished.

However, if one or more fragments have not been received as per the ACK (i.e. the corresponding bits are not set in the Bitmap) then the sender resends the missing fragments. When all missing fragments have been retransmitted, the sender starts the Retransmission Timer (even if an All-0 or an All-1 has not been sent during the retransmission) and waits for an ACK. Upon receipt of the ACK, if one or more fragments have not yet been received, the counter Attempts is increased and the sender resends the missing fragments again. When Attempts reaches MAX_ACK_REQUESTS, the sender aborts the on-going fragmented packet transmission by sending an Abort message and releases any resources for transmission of the packet. The sender also aborts an on-going fragmented packet transmission when a failed MIC check is reported by the receiver.

On the other hand, at the beginning, the receiver side expects to receive window 0. Any fragment received but not belonging to the current window is discarded. All fragments belonging to the correct window are accepted, and the actual fragment number managed by the receiver is computed based on the FCN value. The receiver prepares the Bitmap to report the correctly received and the missing fragments for the current window. After each fragment is received the receiver initializes the Inactivity timer, if the Inactivity Timer expires the transmission is aborted.

When an All-0 fragment is received, it indicates that all the fragments have been sent in the current window. Since the sender is not obliged to always send a full window, some fragment number not set in the receiver memory may not correspond to losses. The receiver sends the corresponding ACK, the Inactivity Timer is set and the transmission of the next window by the sender can start.

If an All-0 fragment has been received and all fragments of the current window have also been received, the receiver then expects a new Window and waits for the next fragment. Upon receipt of a fragment, if the window value has not changed, the received fragments are part of a retransmission. A receiver that has already received a fragment should discard it, otherwise, it updates the Bitmap. If all the bits of the Bitmap are set to one, the receiver may send an ACK

without waiting for an All-0 fragment and the Inactivity Timer is initialized.

On the other hand, if the window value of the next received fragment is set to the next expected window value, this means that the sender has received a correct Bitmap reporting that all fragments have been received. The receiver then updates the value of the next expected window.

If the receiver receives an All-0 fragment, the sender may send one or more fragments per window. Otherwise, some fragments in the window have been lost.

When an All-1 fragment is received, it indicates that the last fragment of the packet has been sent. Since the last window is not always full, the MIC will be used to detect if all fragments of the packet have been received. A correct MIC indicates the end of the transmission but the receiver must stay alive for an Inactivity Timer period to answer to any empty All-1 fragments the sender may send if ACKs sent by the receiver are lost. If the MIC is incorrect, some fragments have been lost. The receiver sends the ACK regardless of successful fragmented packet reception or not, the Inactivity Timer is set. In case of an incorrect MIC, the receiver waits for fragments belonging to the same window. After MAX_ACK_REQUESTS, the receiver will abort the on-going fragmented packet transmission. The receiver also Aborts upon Inactivity Timer expiration.

5.5.2.2. ACK-on-error

The ACK-on-error sender is similar to ACK-Always, the main difference being that in ACK-on-error the ACK is not sent at the end of each window but only when at least one fragment of the current window has been lost (with the exception of the last window, see next paragraph). In Ack-on-error, the Retransmission Timer expiration will be considered as a positive acknowledgment. The Retransmission Timer is set when sending an All-0 or an All-1 fragment. When the All-1 fragment has been sent, then the on-going fragmented packet transmission fragmentation is finished and the sender waits for the last ACK. At the receiver side, when the All-1 fragment is sent and the MIC check indicates successful packet reception, an ACK is also sent to confirm the end of a correct transmission. If the Retransmission Timer expires, an All-1 empty request for the last ACK MUST be sent by the sender to complete the fragmented packet transmission.

If the sender receives an ACK, it checks the window value. ACKs with an unexpected window number are discarded. If the window number on the received Bitmap is correct, the sender verifies if the receiver

has received all fragments of the current window. When at least one fragment has been lost, the counter Attempts is increased by one and the sender resends the missing fragments again. When Attempts reaches MAX_ACK_REQUESTS, the sender sends an Abort message and releases all resources for the on-going fragmented packet transmission. When the retransmission of missing fragments is finished, the sender starts listening for an ACK (even if an All-0 or an All-1 has not been sent during the retransmission) and initializes and starts the Retransmission Timer. After sending an All-1 fragment, the sender listens for an ACK, initializes Attempts, and initializes and starts the Retransmission Timer. If the Retransmission Timer expires, Attempts is increased by one and an empty All-1 fragment is sent to request the ACK for the last window. If Attempts reaches MAX_ACK_REQUESTS, the on-going fragmented packet transmission is aborted.

Unlike the sender, the receiver for ACK-on-error has a larger amount of differences compared with ACK-Always. First, an ACK is not sent unless there is a lost fragment or an unexpected behavior (with the exception of the last window, where an ACK is always sent regardless of fragment losses or not). The receiver starts by expecting fragments from window 0 and maintains the information regarding which fragments it receives. After receiving a fragment, the Inactivity Timer is set, if no fragment has been received and the Inactivity Timer expires the transmission is aborted.

Any fragment not belonging to the current window is discarded. The actual fragment number is computed based on the FCN value. When an All-0 fragment is received and all fragments have been received, the receiver updates the expected window value.

If an All-0 fragment is received, even if another fragment is missing, all fragments from the current window have been sent. Since the sender is not obligated to send a full window, a fragment number not used may not necessarily correspond to losses. As the receiver does not know if the missing fragments are lost or not, it sends an ACK and reinitialises the Inactivity Timer.

On the other hand, after receiving an All-0 fragment, the receiver expects a new window and waits for the next fragment. If the window value of the next fragment has not changed, the received fragment is a retransmission. A receiver that has already received a fragment should discard it. If all fragments of a window (that is not the last one) have been received, the receiver does not send an ACK. While the receiver waits for the next window and if the window value is set to the next value, and if an All-1 fragment with the next value window arrived the receiver aborts the on-going

fragmented packet transmission, and it drops the fragments of the aborted packet transmission.

If the receiver receives an All-1 fragment, this means that the transmission should be finished. If the MIC is incorrect some fragments have been lost. Regardless of fragment losses, the receiver sends an ACK and initializes the Inactivity Timer.

Reception of an All-1 fragment indicates the last fragment of the packet has been sent. Since the last window is not always full, the MIC will be used to detect if all fragments of the window have been received. A correct MIC check indicates the end of the fragmented packet transmission. An ACK is sent by the fragment receiver. In case of an incorrect MIC, the receiver waits for fragments belonging to the same window or the expiration of the Inactivity Timer. The latter will lead the receiver to abort the on-going fragmented packet transmission.

5.5.3. Bitmap Optimization

The Bitmap is transmitted by a receiver as part of the ACK format when there are some missing fragments in a window. An ACK message may introduce padding at the end to align transmitted data to a byte boundary. The first byte boundary includes one or more complete bytes, depending on the size of Rule ID and DTag.

Note that the ACK sent in response to an All-1 fragment includes the C bit. Therefore, the window size and thus the Bitmap size need to be determined taking into account the available space in the layer two frame payload, where there will be 1 bit less for an ACK sent in response to an All-1 fragment than in other ACKs.

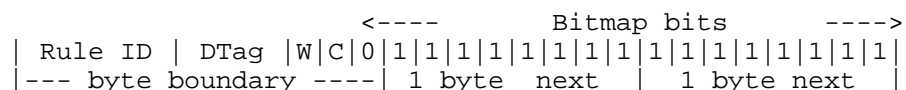


Figure 18: Bitmap

The Bitmap, when transmitted, MUST be optimized in size to reduce the resulting frame size. The right-most bytes with all Bitmap bits set to 1 MUST NOT be transmitted. As the receiver knows the Bitmap size, it can reconstruct the original Bitmap without this optimization. In the example Figure 19, the last 2 bytes of the Bitmap shown in Figure 18 comprise all bits set to 1, therefore, the last 2 bytes of the Bitmap are not sent.

In the last window, when checked bit C value is 1, it means that the received MIC matches the one computed by the receiver, and thus the Bitmap is not sent. Otherwise, the Bitmap needs to be sent after the C bit. Note that the introduction of a C bit may force to reduce the number of fragments in a window to allow the bitmap to fit in a frame.

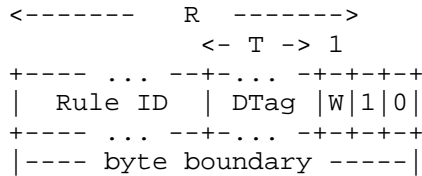


Figure 19: Bitmap transmitted fragment format

Figure 20 shows an example of an ACK (for N=3), where the Bitmap indicates that the second and the fifth fragments have not been correctly received.

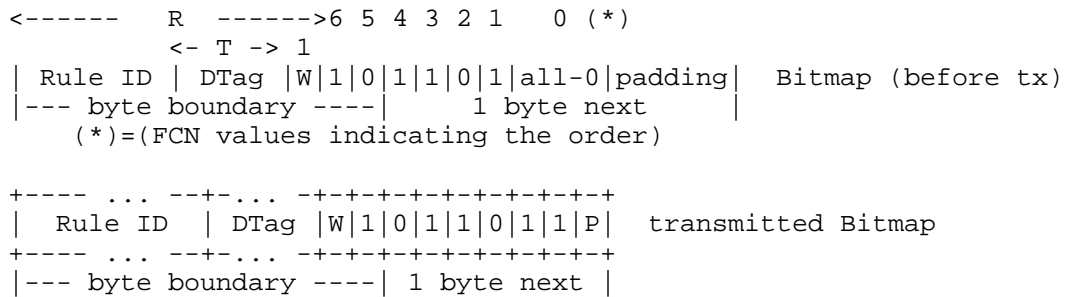


Figure 20: Example of a Bitmap before transmission, and the transmitted one, in any window except the last one, for N=3

Figure 21 shows an example of an ACK (for N=3), where the Bitmap indicates that the MIC check has failed but there are no missing fragments.

```

<----- R -----> 6 5 4 3 2 1 7 (*)
      <- T -> 1 1
| Rule ID | DTag |W|0|1|1|1|1|1|1|padding| Bitmap (before tx)
|---- byte boundary ----| 1 byte next | 1 byte next |
      C
+----- ... ---+... -+++++
| Rule ID | DTag |W|0|1| transmitted Bitmap
+----- ... ---+... -+++++
|---- byte boundary ----|
      (*) = (FCN values indicating the order)

```

Figure 21: Example of the Bitmap in Window mode for the last window, for N=3)

5.6. Supporting multiple window sizes

For ACK-Always or ACK-on-error, implementers may opt to support a single window size or multiple window sizes. The latter, when feasible, may provide performance optimizations. For example, a large window size may be used for packets that need to be carried by a large number of fragments. However, when the number of fragments required to carry a packet is low, a smaller window size, and thus a shorter Bitmap, may be sufficient to provide feedback on all fragments. If multiple window sizes are supported, the Rule ID may be used to signal the window size in use for a specific packet transmission.

Note that the same window size MUST be used for the transmission of all fragments that belong to a packet.

5.7. Downlink fragment transmission

In some LPWAN technologies, as part of energy-saving techniques, downlink transmission is only possible immediately after an uplink transmission. In order to avoid potentially high delay for fragmented datagram transmission in the downlink, the fragment receiver MAY perform an uplink transmission as soon as possible after reception of a fragment that is not the last one. Such uplink transmission may be triggered by the L2 (e.g. an L2 ACK sent in response to a fragment encapsulated in a L2 frame that requires an L2 ACK) or it may be triggered from an upper layer.

For fragmented packet transmission in the downlink, and when ACK Always is used, the fragment receiver MAY support timer-based ACK retransmission. In this mechanism, the fragment receiver initializes and starts a timer (the Inactivity Timer is used) after the transmission of an ACK, except when the ACK is sent in response to

the last fragment of a packet (All-1 fragment). In the latter case, the fragment receiver does not start a timer after transmission of the ACK.

If, after transmission of an ACK that is not an All-1 fragment, and before expiration of the corresponding Inactivity timer, the fragment receiver receives a fragment that belongs to the current window (e.g. a missing fragment from the current window) or to the next window, the Inactivity timer for the ACK is stopped. However, if the Inactivity timer expires, the ACK is resent and the Inactivity timer is reinitialized and restarted.

The default initial value for the Inactivity timer, as well as the maximum number of retries for a specific ACK, denoted `MAX_ACK_RETRIES`, are not defined in this document, and need to be defined in other documents (e.g. technology-specific profiles). The initial value of the Inactivity timer is expected to be greater than that of the Retransmission timer, in order to make sure that a (buffered) fragment to be retransmitted can find an opportunity for that transmission.

When the fragment sender transmits the All-1 fragment, it initializes and starts its retransmission timer to a long value (e.g. several times the initial Inactivity timer). If an ACK is received before expiration of this timer, the fragment sender retransmits any lost fragments reported by the ACK, or if the ACK confirms successful reception of all fragments of the last window, transmission of the fragmented packet ends. If the timer expires, and no ACK has been received since the start of the timer, the fragment sender assumes that the All-1 fragment has been successfully received (and possibly, the last ACK has been lost: this mechanism assumes that the retransmission timer for the All-1 fragment is long enough to allow several ACK retries if the All-1 fragment has not been received by the fragment receiver, and it also assumes that it is unlikely that several ACKs become all lost).

6. Padding management

SCHC header, either for compression, fragmentation or acknowledgment does not preserve byte alignment. Since most of the LPWAN network technologies payload is expressed in an integer number of bytes; the sender will introduce at the end some padding bits while the receiver must be able to eliminate them.

The algorithm for padding bit elimination for compressed or fragmented frames is simple. Based on the following principle: * The SCHC header is not aligned on a byte boundary, but its size in bits is given by the rule.

- o The data size is variable, but always a multiple of 8 bits.
- o Padding bits MUST never exceed 7 bits.

In that case, a receiver after decoding the SCHC header, must take the maximum multiple of 8 bits as data. The remaining bits are padding bits.

7. SCHC Compression for IPv6 and UDP headers

This section lists the different IPv6 and UDP header fields and how they can be compressed.

7.1. IPv6 version field

This field always holds the same value. Therefore, the TV is 6, the MO is "equal" and the CDA "not-sent".

7.2. IPv6 Traffic class field

If the DiffServ field identified by the rest of the rule does not vary and is known by both sides, the TV should contain this well-known value, the MO should be "equal" and the CDA must be "not-sent".

If the DiffServ field identified by the rest of the rule varies over time or is not known by both sides, then there are two possibilities depending on the variability of the value: The first one is to do not compressed the field and sends the original value. In the second, where the values can be computed by sending only the LSB bits:

- o TV is not set to any value, MO is set to "ignore" and CDA is set to "value-sent"
- o TV contains a stable value, MO is MSB(X) and CDA is set to LSB

7.3. Flow label field

If the Flow Label field identified by the rest of the rule does not vary and is known by both sides, the TV should contain this well-known value, the MO should be "equal" and the CDA should be "not-sent".

If the Flow Label field identified by the rest of the rule varies during time or is not known by both sides, there are two possibilities depending on the variability of the value: The first one is without compression and then the value is sent. In the second, only part of the value is sent and the decompressor needs to compute the original value:

- o TV is not set, MO is set to "ignore" and CDA is set to "value-sent"
- o TV contains a stable value, MO is MSB(X) and CDA is set to LSB

7.4. Payload Length field

If the LPWAN technology does not add padding, this field can be elided for the transmission on the LPWAN network. The SCHC C/D recomputes the original payload length value. The TV is not set, the MO is set to "ignore" and the CDA is "compute-IPv6-length".

If the payload length needs to be sent and does not need to be coded in 16 bits, the TV can be set to 0x0000, the MO set to "MSB (16-s)" and the CDA to "LSB". The 's' parameter depends on the expected maximum packet length.

In other cases, the payload length field must be sent and the CDA is replaced by "value-sent".

7.5. Next Header field

If the Next Header field identified by the rest of the rule does not vary and is known by both sides, the TV should contain this Next Header value, the MO should be "equal" and the CDA should be "not-sent".

If the Next Header field identified by the rest of the rule varies during time or is not known by both sides, then TV is not set, MO is set to "ignore" and CDA is set to "value-sent". A matching-list may also be used.

7.6. Hop Limit field

The End System is generally a device and does not forward packets. Therefore, the Hop Limit value is constant. So, the TV is set with a default value, the MO is set to "equal" and the CDA is set to "not-sent".

Otherwise the value is sent on the LPWAN: TV is not set, MO is set to ignore and CDA is set to "value-sent".

Note that the field behavior differs in upstream and downstream. In upstream, since there is no IP forwarding between the Dev and the SCHC C/D, the value is relatively constant. On the other hand, the downstream value depends of Internet routing and may change more frequently. One solution could be to use the Direction Indicator

(DI) to distinguish both directions to elide the field in the upstream direction and send the value in the downstream direction.

7.7. IPv6 addresses fields

As in 6LoWPAN [RFC4944], IPv6 addresses are splitted into two 64-bit long fields; one for the prefix and one for the Interface Identifier (IID). These fields should be compressed. To allow a single rule, these values are identified by their role (DEV or APP) and not by their position in the frame (source or destination). The SCHC C/D must be aware of the traffic direction (upstream, downstream) to select the appropriate field.

7.7.1. IPv6 source and destination prefixes

Both ends must be synchronized with the appropriate prefixes. For a specific flow, the source and destination prefixes can be unique and stored in the context. It can be either a link-local prefix or a global prefix. In that case, the TV for the source and destination prefixes contain the values, the MO is set to "equal" and the CDA is set to "not-sent".

In case the rule allows several prefixes, mapping-list must be used. The different prefixes are listed in the TV associated with a short ID. The MO is set to "match-mapping" and the CDA is set to "mapping-sent".

Otherwise the TV contains the prefix, the MO is set to "equal" and the CDA is set to "value-sent".

7.7.2. IPv6 source and destination IID

If the DEV or APP IID are based on an LPWAN address, then the IID can be reconstructed with information coming from the LPWAN header. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "DEViid" or "APPiid". Note that the LPWAN technology is generally carrying a single device identifier corresponding to the DEV. The SCHC C/D may also not be aware of these values.

If the DEV address has a static value that is not derived from an IEEE EUI-64, then TV contains the actual Dev address value, the MO operator is set to "equal" and the CDA is set to "not-sent".

If several IIDs are possible, then the TV contains the list of possible IIDs, the MO is set to "match-mapping" and the CDA is set to "mapping-sent".

Otherwise the value variation of the IID may be reduced to few bytes. In that case, the TV is set to the stable part of the IID, the MO is set to "MSB" and the CDA is set to "LSB".

Finally, the IID can be sent on the LPWAN. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "value-sent".

7.8. IPv6 extensions

No extension rules are currently defined. They can be based on the MOs and CDAs described above.

7.9. UDP source and destination port

To allow a single rule, the UDP port values are identified by their role (DEV or APP) and not by their position in the frame (source or destination). The SCHC C/D must be aware of the traffic direction (upstream, downstream) to select the appropriate field. The following rules apply for DEV and APP port numbers.

If both ends know the port number, it can be elided. The TV contains the port number, the MO is set to "equal" and the CDA is set to "not-sent".

If the port variation is on few bits, the TV contains the stable part of the port number, the MO is set to "MSB" and the CDA is set to "LSB".

If some well-known values are used, the TV can contain the list of these values, the MO is set to "match-mapping" and the CDA is set to "mapping-sent".

Otherwise the port numbers are sent on the LPWAN. The TV is not set, the MO is set to "ignore" and the CDA is set to "value-sent".

7.10. UDP length field

If the LPWAN technology does not introduce padding, the UDP length can be computed from the received data. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "compute-UDP-length".

If the payload is small, the TV can be set to 0x0000, the MO set to "MSB" and the CDA to "LSB".

On other cases, the length must be sent and the CDA is replaced by "value-sent".

7.11. UDP Checksum field

IPv6 mandates a checksum in the protocol above IP. Nevertheless, if a more efficient mechanism such as L2 CRC or MIC is carried by or over the L2 (such as in the LPWAN fragmentation process (see Section 5)), the UDP checksum transmission can be avoided. In that case, the TV is not set, the MO is set to "ignore" and the CDA is set to "compute-UDP-checksum".

In other cases, the checksum must be explicitly sent. The TV is not set, the MO is set to "ignore" and the CDF is set to "value-sent".

8. Security considerations

8.1. Security considerations for header compression

A malicious header compression could cause the reconstruction of a wrong packet that does not match with the original one, such corruption may be detected with end-to-end authentication and integrity mechanisms. Denial of Service may be produced but its arise other security problems that may be solved with or without header compression.

8.2. Security considerations for fragmentation

This subsection describes potential attacks to LPWAN fragmentation and suggests possible countermeasures.

A node can perform a buffer reservation attack by sending a first fragment to a target. Then, the receiver will reserve buffer space for the IPv6 packet. Other incoming fragmented packets will be dropped while the reassembly buffer is occupied during the reassembly timeout. Once that timeout expires, the attacker can repeat the same procedure, and iterate, thus creating a denial of service attack. The (low) cost to mount this attack is linear with the number of buffers at the target node. However, the cost for an attacker can be increased if individual fragments of multiple packets can be stored in the reassembly buffer. To further increase the attack cost, the reassembly buffer can be splitted into fragment-sized buffer slots. Once a packet is complete, it is processed normally. If buffer overload occurs, a receiver can discard packets based on the sender behavior, which may help identify which fragments have been sent by an attacker.

In another type of attack, the malicious node is required to have overhearing capabilities. If an attacker can overhear a fragment, it can send a spoofed duplicate (e.g. with random payload) to the destination. If the LPWAN technology does not support suitable

protection (e.g. source authentication and frame counters to prevent replay attacks), a receiver cannot distinguish legitimate from spoofed fragments. Therefore, the original IPv6 packet will be considered corrupt and will be dropped. To protect resource-constrained nodes from this attack, it has been proposed to establish a binding among the fragments to be transmitted by a node, by applying content-chaining to the different fragments, based on cryptographic hash functionality. The aim of this technique is to allow a receiver to identify illegitimate fragments.

Further attacks may involve sending overlapped fragments (i.e. comprising some overlapping parts of the original IPv6 datagram). Implementers should make sure that the correct operation is not affected by such event.

In Window mode - ACK on error, a malicious node may force a fragment sender to resend a fragment a number of times, with the aim to increase consumption of the fragment sender's resources. To this end, the malicious node may repeatedly send a fake ACK to the fragment sender, with a Bitmap that reports that one or more fragments have been lost. In order to mitigate this possible attack, `MAX_FRAG_RETRIES` may be set to a safe value which allows to limit the maximum damage of the attack to an acceptable extent. However, note that a high setting for `MAX_FRAG_RETRIES` benefits fragment delivery reliability, therefore the trade-off needs to be carefully considered.

9. Acknowledgements

Thanks to Dominique Barthel, Carsten Bormann, Philippe Clavier, Eduardo Ingles Sanchez, Arunprabhu Kandasamy, Sergio Lopez Bernal, Antony Markovski, Alexander Pelov, Pascal Thubert, Juan Carlos Zuniga and Diego Dujovne for useful design consideration and comments.

10. References

10.1. Normative References

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [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, <<https://www.rfc-editor.org/info/rfc4944>>.

- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <<https://www.rfc-editor.org/info/rfc5795>>.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <<https://www.rfc-editor.org/info/rfc7136>>.

10.2. Informative References

- [I-D.ietf-lpwan-overview]
Farrell, S., "LPWAN Overview", draft-ietf-lpwan-overview-07 (work in progress), October 2017.

Appendix A. SCHC Compression Examples

This section gives some scenarios of the compression mechanism for IPv6/UDP. The goal is to illustrate the SCHC behavior.

The most common case using the mechanisms defined in this document will be a LPWAN Dev that embeds some applications running over CoAP. In this example, three flows are considered. The first flow is for the device management based on CoAP using Link Local IPv6 addresses and UDP ports 123 and 124 for Dev and App, respectively. The second flow will be a CoAP server for measurements done by the Device (using ports 5683) and Global IPv6 Address prefixes alpha::IID/64 to beta::1/64. The last flow is for legacy applications using different ports numbers, the destination IPv6 address prefix is gamma::1/64.

Figure 22 presents the protocol stack for this Device. IPv6 and UDP are represented with dotted lines since these protocols are compressed on the radio link.

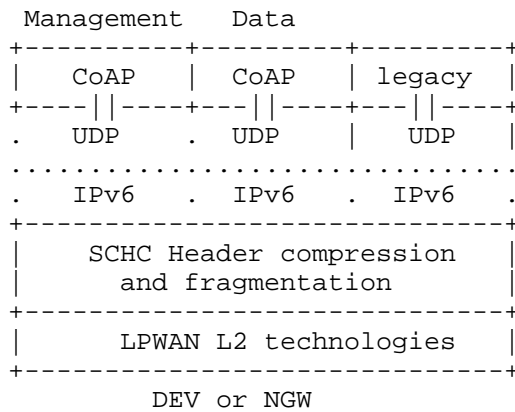


Figure 22: Simplified Protocol Stack for LP-WAN

Note that in some LPWAN technologies, only the Devs have a device ID. Therefore, when such technologies are used, it is necessary to define statically an IID for the Link Local address for the SCHC C/D.

Rule 0

Field	FL	FP	DI	Value	Match Opera.	Comp Decomp Action	Sent [bits]
IPv6 version	4	1	Bi	6	equal	not-sent	
IPv6 DiffServ	8	1	Bi	0	equal	not-sent	
IPv6 Flow Label	20	1	Bi	0	equal	not-sent	
IPv6 Length	16	1	Bi		ignore	comp-length	
IPv6 Next Header	8	1	Bi	17	equal	not-sent	
IPv6 Hop Limit	8	1	Bi	255	ignore	not-sent	
IPv6 DEVprefix	64	1	Bi	FE80::/64	equal	not-sent	
IPv6 DEViid	64	1	Bi		ignore	DEViid	
IPv6 APPprefix	64	1	Bi	FE80::/64	equal	not-sent	
IPv6 APPiid	64	1	Bi	::1	equal	not-sent	
UDP DEVport	16	1	Bi	123	equal	not-sent	
UDP APPport	16	1	Bi	124	equal	not-sent	
UDP Length	16	1	Bi		ignore	comp-length	
UDP checksum	16	1	Bi		ignore	comp-chk	

Rule 1

Field	FL	FP	DI	Value	Match Opera.	Action	Sent [bits]
						Action	

IPv6 version	4	1	Bi	6	equal	not-sent	
IPv6 DiffServ	8	1	Bi	0	equal	not-sent	
IPv6 Flow Label	20	1	Bi	0	equal	not-sent	
IPv6 Length	16	1	Bi		ignore	comp-length	
IPv6 Next Header	8	1	Bi	17	equal	not-sent	
IPv6 Hop Limit	8	1	Bi	255	ignore	not-sent	
IPv6 DEVprefix	64	1	Bi	[alpha/64, fe80::<64]	match- mapping	mapping-sent	[1]
IPv6 DEViid	64	1	Bi		ignore	DEViid	
IPv6 APPprefix	64	1	Bi	[beta/64, alpha/64, fe80::<64]	match- mapping	mapping-sent	[2]
IPv6 APPiid	64	1	Bi	::1000	equal	not-sent	
UDP DEVport	16	1	Bi	5683	equal	not-sent	
UDP APPport	16	1	Bi	5683	equal	not-sent	
UDP Length	16	1	Bi		ignore	comp-length	
UDP checksum	16	1	Bi		ignore	comp-chk	

Rule 2

Field	FL	FP	DI	Value	Match Opera.	Action Action	Sent [bits]
IPv6 version	4	1	Bi	6	equal	not-sent	
IPv6 DiffServ	8	1	Bi	0	equal	not-sent	
IPv6 Flow Label	20	1	Bi	0	equal	not-sent	
IPv6 Length	16	1	Bi		ignore	comp-length	
IPv6 Next Header	8	1	Bi	17	equal	not-sent	
IPv6 Hop Limit	8	1	Up	255	ignore	not-sent	
IPv6 Hop Limit	8	1	Dw		ignore	value-sent	[8]
IPv6 DEVprefix	64	1	Bi	alpha/64	equal	not-sent	
IPv6 DEViid	64	1	Bi		ignore	DEViid	
IPv6 APPprefix	64	1	Bi	gamma/64	equal	not-sent	
IPv6 APPiid	64	1	Bi	::1000	equal	not-sent	
UDP DEVport	16	1	Bi	8720	MSB(12)	LSB(4)	[4]
UDP APPport	16	1	Bi	8720	MSB(12)	LSB(4)	[4]
UDP Length	16	1	Bi		ignore	comp-length	
UDP checksum	16	1	Bi		ignore	comp-chk	

Figure 23: Context rules

All the fields described in the three rules depicted on Figure 23 are present in the IPv6 and UDP headers. The DEViid-DID value is found in the L2 header.

The second and third rules use global addresses. The way the Dev learns the prefix is not in the scope of the document.

The third rule compresses port numbers to 4 bits.

Appendix B. Fragmentation Examples

This section provides examples of different fragment delivery reliability options possible on the basis of this specification.

Figure 24 illustrates the transmission of an IPv6 packet that needs 11 fragments in the No ACK option. Where FCN is always 1 bit.

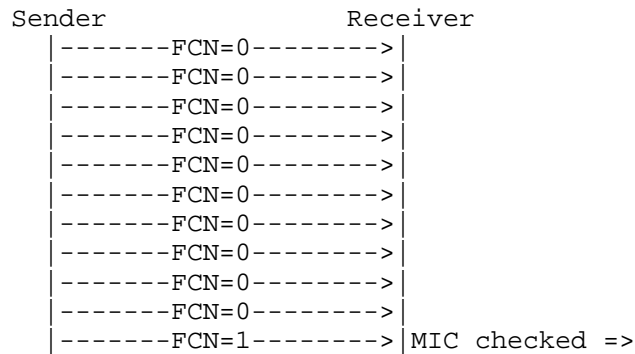


Figure 24: Transmission of an IPv6 packet carried by 11 fragments in the No ACK option

Figure 25 illustrates the transmission of an IPv6 packet that needs 11 fragments in ACK-on-error, for N=3, without losses.

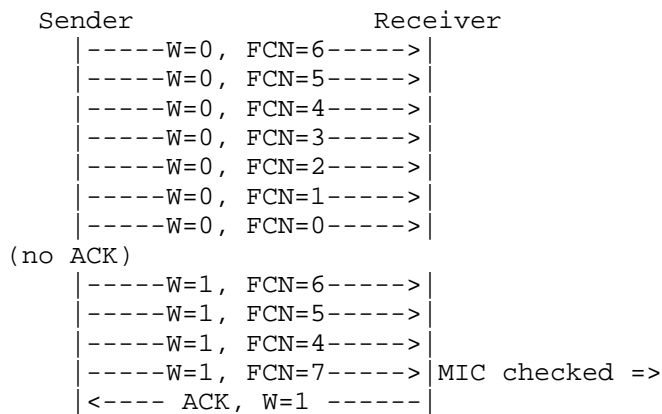


Figure 25: Transmission of an IPv6 packet carried by 11 fragments in ACK-on-error, for N=3 and MAX_WIND_FCN=6, without losses.

Figure 26 illustrates the transmission of an IPv6 packet that needs 11 fragments ACK-on-error, for N=3, with three losses.

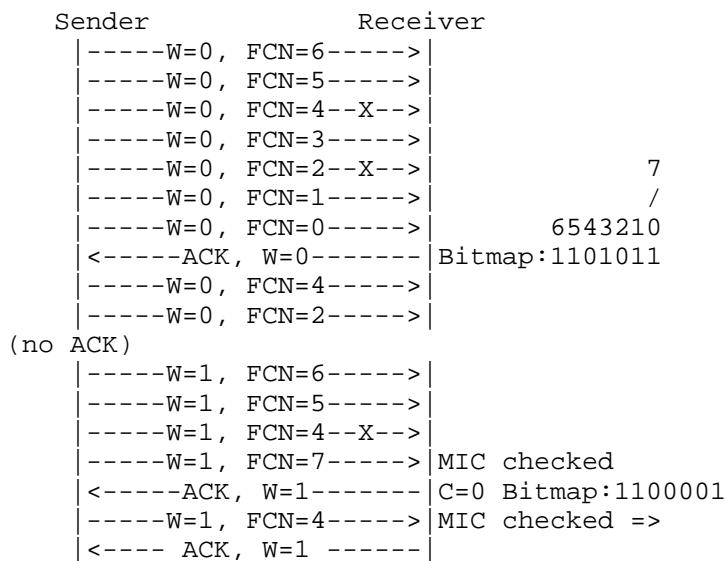


Figure 26: Transmission of an IPv6 packet carried by 11 fragments in ACK-on-error, for N=3 and MAX_WIND_FCN=6, three losses.

Figure 27 illustrates the transmission of an IPv6 packet that needs 11 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, without

losses. Note: in Window mode, an additional bit will be needed to number windows.

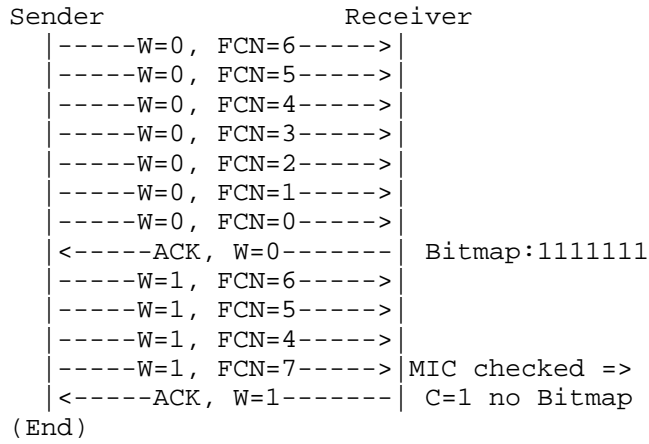


Figure 27: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, no losses.

Figure 28 illustrates the transmission of an IPv6 packet that needs 11 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, with three losses.

```

Sender                               Receiver
|-----W=1, FCN=6----->|
|-----W=1, FCN=5----->|
|-----W=1, FCN=4--X-->|
|-----W=1, FCN=3----->|
|-----W=1, FCN=2--X-->|
|-----W=1, FCN=1----->|
|-----W=1, FCN=0----->|
|<-----ACK, W=1-----|
|-----W=1, FCN=4----->|
|-----W=1, FCN=2----->|
|<-----ACK, W=1-----|
|-----W=0, FCN=6----->|
|-----W=0, FCN=5----->|
|-----W=0, FCN=4--X-->|
|-----W=0, FCN=7----->|
|<-----ACK, W=0-----|
|-----W=0, FCN=4----->|
|<-----ACK, W=0-----|
(End)

```

7
/
6543210
Bitmap:1101011
Bitmap:
MIC checked
C= 0 Bitmap:11000001
MIC checked =>
C= 1 no Bitmap

Figure 28: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses.

Figure 29 illustrates the transmission of an IPv6 packet that needs 6 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, with three losses, and only one retry is needed for each lost fragment. Note that, since a single window is needed for transmission of the IPv6 packet in this case, the example illustrates behavior when losses happen in the last window.

```

Sender                                     Receiver
|-----W=0, CFN=6----->|
|-----W=0, CFN=5----->|
|-----W=0, CFN=4--X-->|
|-----W=0, CFN=3--X-->|
|-----W=0, CFN=2--X-->|
|-----W=0, CFN=7----->| MIC checked
|<-----ACK, W=0-----| C= 0 Bitmap:1100001
|-----W=0, CFN=4----->| MIC checked: failed
|-----W=0, CFN=3----->| MIC checked: failed
|-----W=0, CFN=2----->| MIC checked: success
|<-----ACK, W=0-----| C=1 no Bitmap
(End)

```

Figure 29: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses, and only one retry is needed for each lost fragment.

Figure 30 illustrates the transmission of an IPv6 packet that needs 6 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, with three losses, and the second ACK is lost. Note that, since a single window is needed for transmission of the IPv6 packet in this case, the example illustrates behavior when losses happen in the last window.

```

Sender                                     Receiver
|-----W=0, CFN=6----->|
|-----W=0, CFN=5----->|
|-----W=0, CFN=4--X-->|
|-----W=0, CFN=3--X-->|
|-----W=0, CFN=2--X-->|
|-----W=0, CFN=7----->| MIC checked
|<-----ACK, W=0-----| C=0 Bitmap:1100001
|-----W=0, CFN=4----->| MIC checked: wrong
|-----W=0, CFN=3----->| MIC checked: wrong
|-----W=0, CFN=2----->| MIC checked: right
|X---ACK, W=0-----| C= 1 no Bitmap
timeout
|-----W=0, CFN=7----->|
|<-----ACK, W=0-----| C= 1 no Bitmap
(End)

```

Figure 30: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for N=3, and MAX_WIND_FCN=6, with three losses, and the second ACK is lost.

Figure 31 illustrates the transmission of an IPv6 packet that needs 6 fragments in ACK-Always, for N=3 and MAX_WIND_FCN=6, with three

losses, and one retransmitted fragment is lost. Note that, since a single window is needed for transmission of the IPv6 packet in this case, the example illustrates behavior when losses happen in the last window.

Sender	Receiver
-----W=0, CFN=6----->	
-----W=0, CFN=5----->	
-----W=0, CFN=4--X-->	
-----W=0, CFN=3--X-->	
-----W=0, CFN=2--X-->	
-----W=0, CFN=7----->	MIC checked
<-----ACK, W=0-----	C=0 Bitmap:110001
-----W=0, CFN=4----->	MIC checked: wrong
-----W=0, CFN=3----->	MIC checked: wrong
-----W=0, CFN=2--X-->	
timeout	
-----W=0, CFN=7----->	All-0 empty
<-----ACK, W=0-----	C=0 Bitmap: 1111101
-----W=0, CFN=2----->	MIC checked: right
<-----ACK, W=0-----	C=1 no Bitmap
(End)	

Figure 31: Transmission of an IPv6 packet carried by 11 fragments in ACK-Always, for $N=3$, and $MAX_WIND_FCN=6$, with three losses, and one retransmitted fragment is lost.

Appendix C illustrates the transmission of an IPv6 packet that needs 28 fragments in ACK-Always, for $N=5$ and $MAX_WIND_FCN=23$, with two losses. Note that $MAX_WIND_FCN=23$ may be useful when the maximum possible Bitmap size, considering the maximum lower layer technology payload size and the value of R , is 3 bytes. Note also that the FCN of the last fragment of the packet is the one with $FCN=31$ (i.e. $FCN=2^N-1$ for $N=5$, or equivalently, all FCN bits set to 1).

```

Sender                Receiver
|-----W=0, CFN=23----->|
|-----W=0, CFN=22----->|
|-----W=0, CFN=21--X-->|
|-----W=0, CFN=20----->|
|-----W=0, CFN=19----->|
|-----W=0, CFN=18----->|
|-----W=0, CFN=17----->|
|-----W=0, CFN=16----->|
|-----W=0, CFN=15----->|
|-----W=0, CFN=14----->|
|-----W=0, CFN=13----->|
|-----W=0, CFN=12----->|
|-----W=0, CFN=11----->|
|-----W=0, CFN=10--X-->|
|-----W=0, CFN=9  ----->|
|-----W=0, CFN=8  ----->|
|-----W=0, CFN=7  ----->|
|-----W=0, CFN=6  ----->|
|-----W=0, CFN=5  ----->|
|-----W=0, CFN=4  ----->|
|-----W=0, CFN=3  ----->|
|-----W=0, CFN=2  ----->|
|-----W=0, CFN=1  ----->|
|-----W=0, CFN=0  ----->|
|                               |lcl-Bitmap:110111111111101111111111
|                               |Bitmap:1101111111111011
|<-----ACK, W=0----->|
|-----W=0, CFN=21----->|
|-----W=0, CFN=10----->|
|<-----ACK, W=0----->|no Bitmap
|-----W=1, CFN=23----->|
|-----W=1, CFN=22----->|
|-----W=1, CFN=21----->|
|-----W=1, CFN=31----->|MIC checked =>
|<-----ACK, W=1----->|no Bitmap
(End)

```

Appendix C. Fragmentation State Machines

The fragmentation state machines of the sender and the receiver in the different reliability options are next in the following figures:

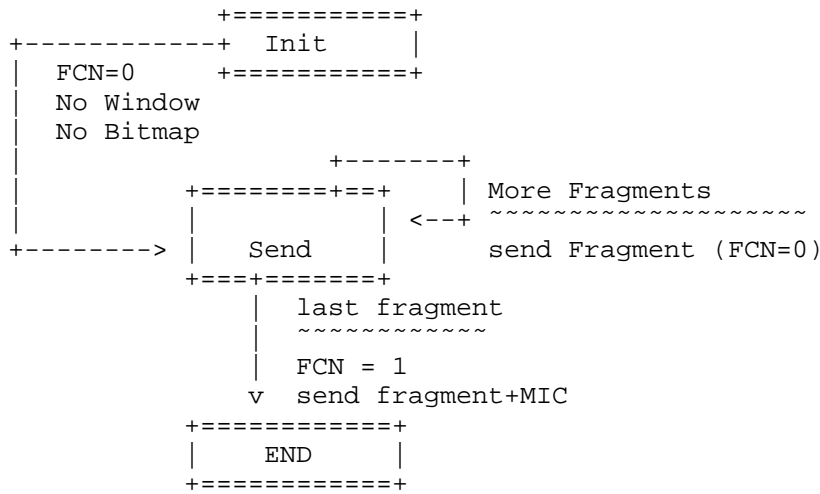


Figure 32: Sender State Machine for the No ACK Mode

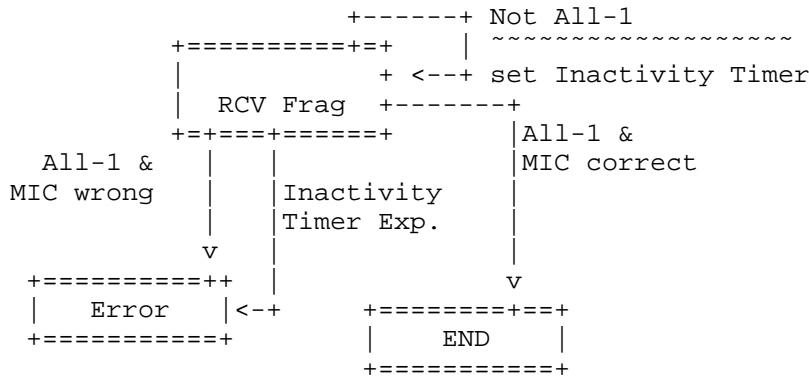


Figure 33: Receiver State Machine for the No ACK Mode

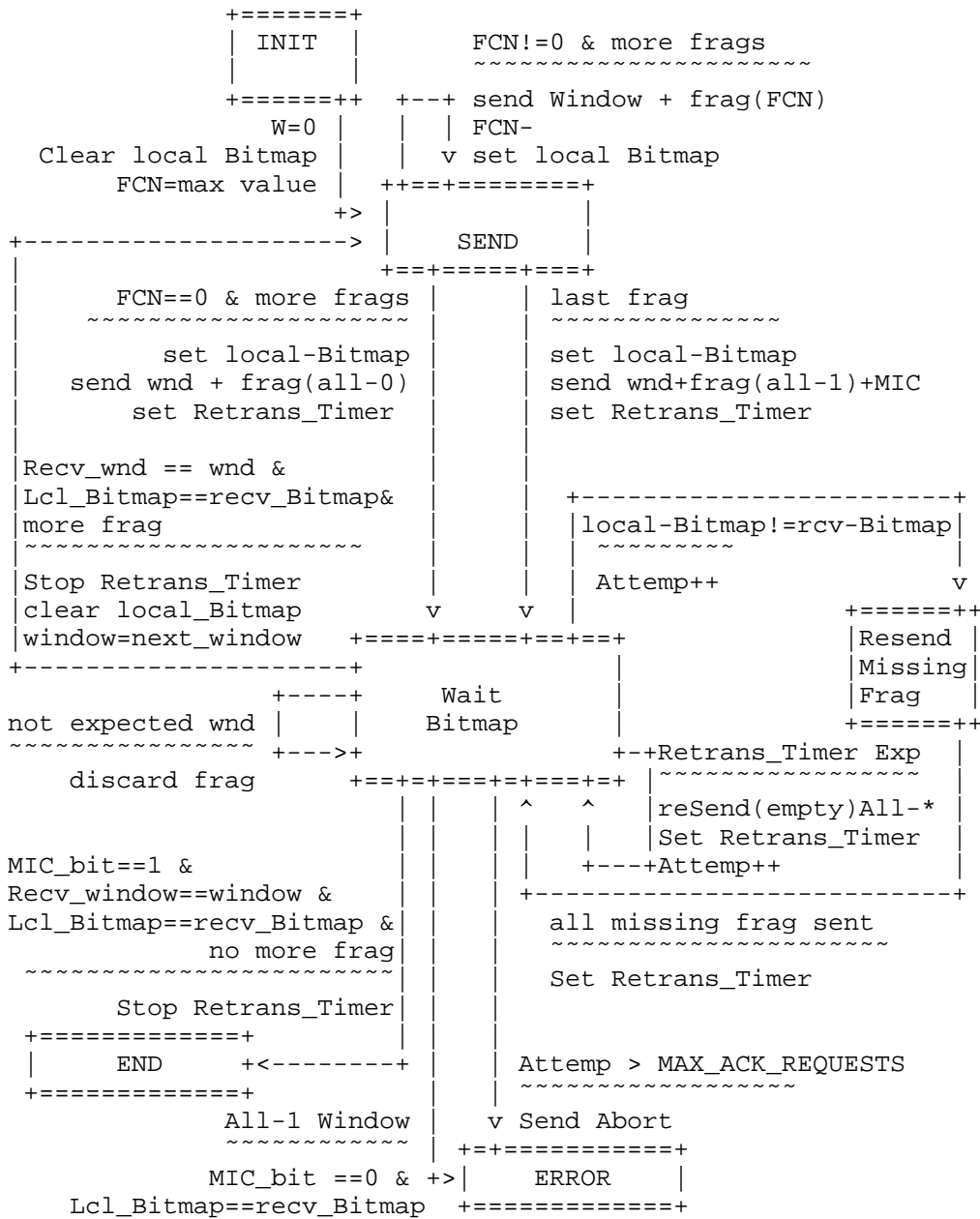
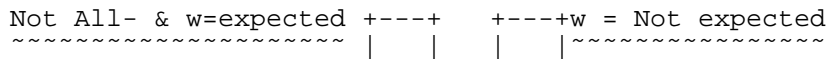


Figure 34: Sender State Machine for the ACK Always Mode




```

Set local_Bitmap(FCN) | v v |discard
+++++=====+
+-----+ Rcv +--->* ABORT
+-----+ Window |
+-----+
All-0 & w=expect | ^ w =next & not-All
~~~~~|~~~~~
set lcl_Bitmap(FCN)| expected = next window
send local_Bitmap | Clear local_Bitmap

w=expct & not-All
~~~~~
set lcl_Bitmap(FCN)+++ | +---+ w=next & All-0
if lcl_Bitmap full | | | | |~~~~~
send lcl_Bitmap | | | | | expct = nxt wnd
v | v v v |
w=expct & All-1 ++++++ | Clear lcl_Bitmap
~~~~~ +->+ Wait +<+ set lcl_Bitmap(FCN)
discard +---| Next | send lcl_Bitmap
All-0 +-----+ Window +--->* ABORT
~~~~~ +----->+++++
snd lcl_bm All-1 & w=next | All-1 & w=nxt
& MIC wrong | & MIC right
~~~~~|~~~~~
set local_Bitmap(FCN) | set lcl_Bitmap(FCN)
send local_Bitmap | send local_Bitmap
+-----+-----+
All-1 & w=expct | v +---+ w=expctd &
& MIC wrong | +-----+ | MIC wrong
~~~~~|~~~~~
set local_Bitmap(FCN) | +<+ |~~~~~
send local_Bitmap | Wait End | set lcl_btmap(FCN)
+----->+ +--->* ABORT
+++++=====+ All-1&MIC wrong
~~~~~|~~~~~
w=expected & MIC right | +---+ send lcl_btmap
~~~~~| |
set local_Bitmap(FCN) | +--+ Not All-1
send local_Bitmap | |~~~~~
discard

All-1 & w=expctd & MIC right | +--+ All-1
~~~~~| v | v | v |~~~~~
set local_Bitmap(FCN) | ++++++ Send lcl_btmap
send local_Bitmap | |
+-----+-----+
| END | +<-----+
+++++=====+
--->* ABORT

```

```
~~~~~  
    Inactivity_Timer = expires  
When DWN_Link  
    IF Inactivity_Timer expires  
        Send DWL Request  
        Attemp++
```

Figure 35: Receiver State Machine for the ACK Always Mode

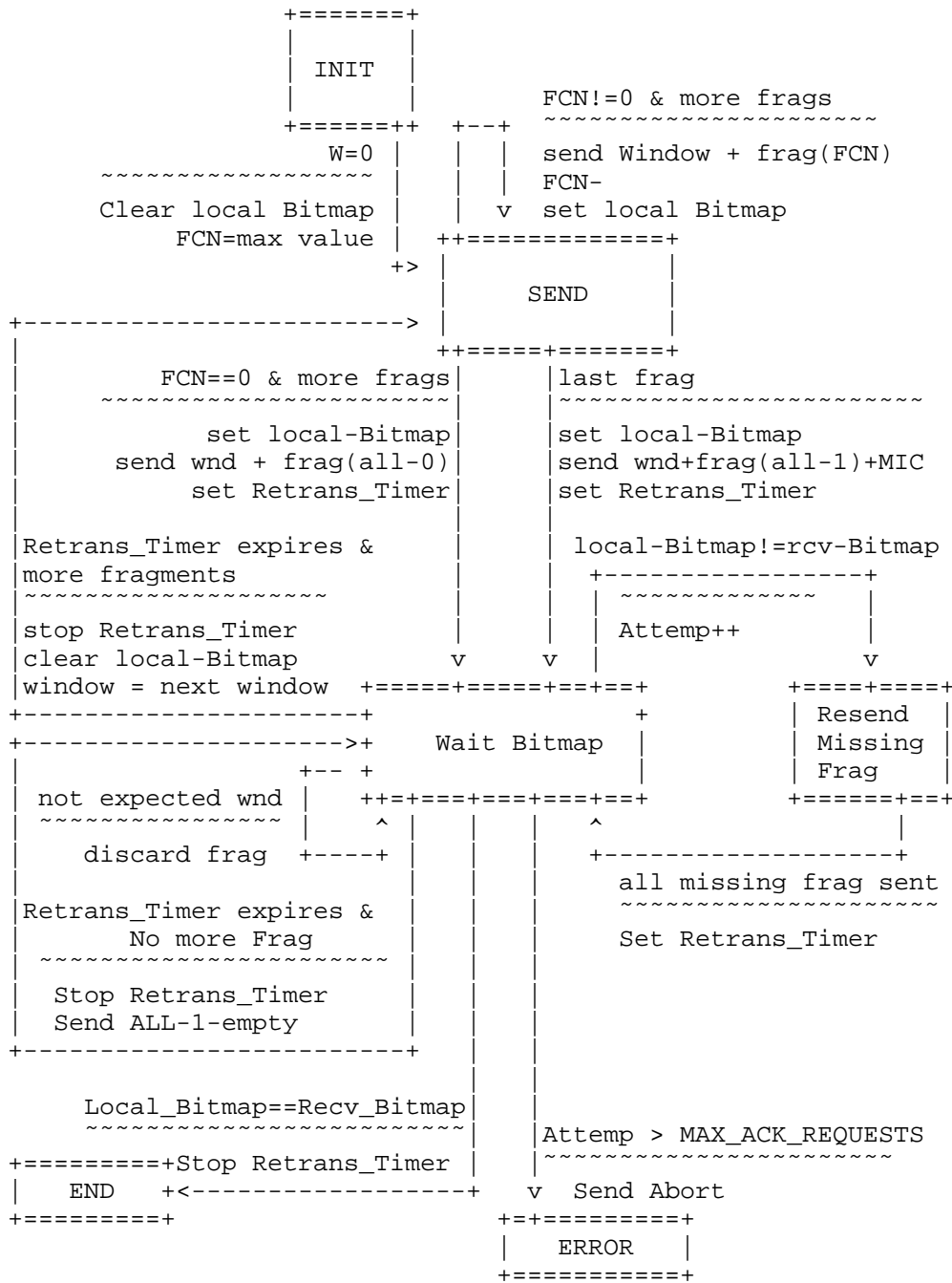


Figure 36: Sender State Machine for the ACK on error Mode

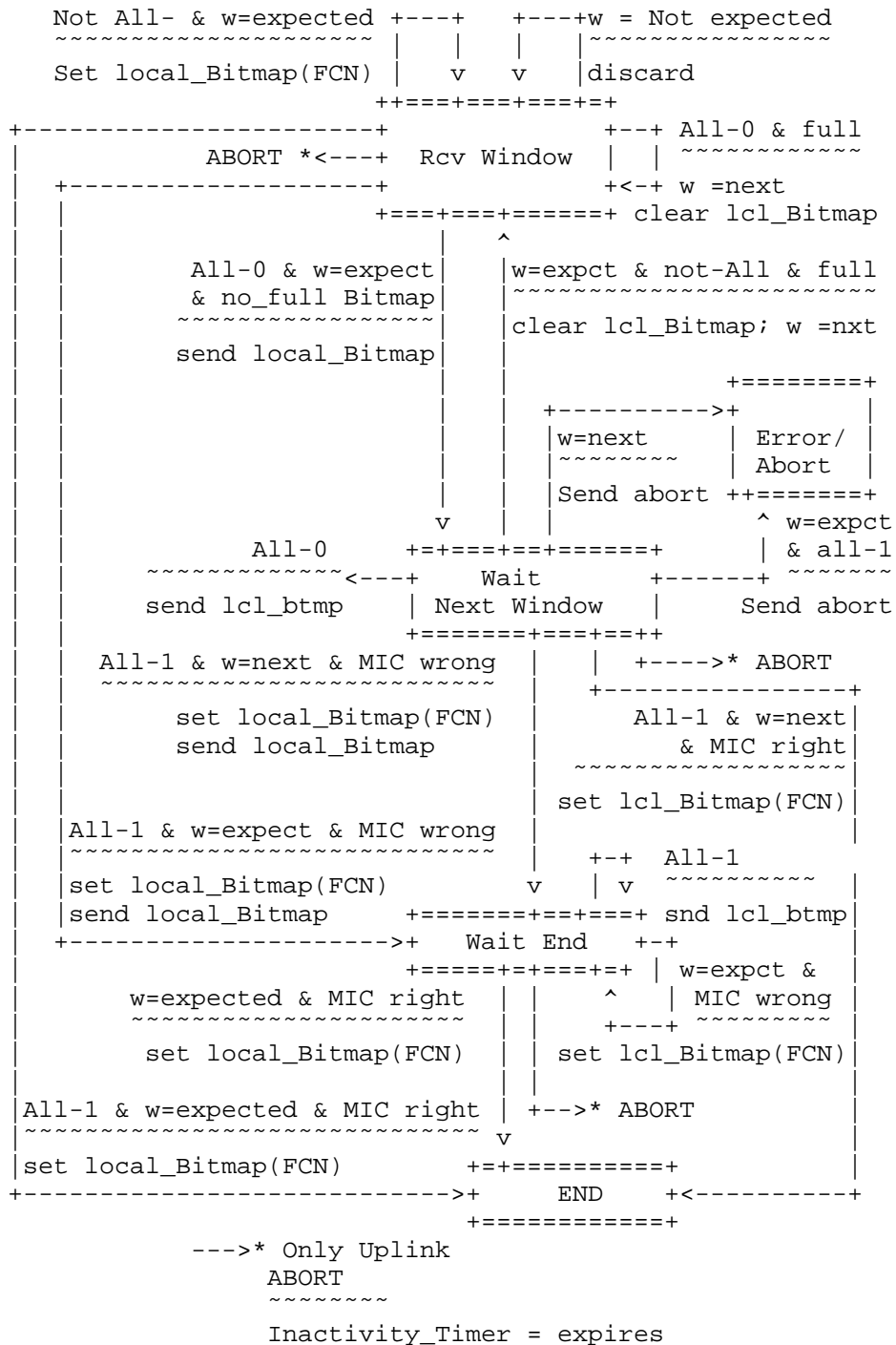


Figure 37: Receiver State Machine for the ACK on error Mode

Appendix D. Allocation of Rule IDs for fragmentation

A set of Rule IDs are allocated to support different aspects of fragmentation functionality as per this document. The allocation of IDs is to be defined in other documents. The set MAY include:

- o one ID or a subset of IDs to identify a fragment as well as its reliability option and its window size, if multiple of these are supported.
- o one ID to identify the ACK message.
- o one ID to identify the Abort message as per Section 9.8.

Appendix E. Note

Carles Gomez has been funded in part by the Spanish Government (Ministerio de Educacion, Cultura y Deporte) through the Jose Castillejo grant CAS15/00336, and by the ERDF and the Spanish Government through project TEC2016-79988-P. Part of his contribution to this work has been carried out during his stay as a visiting scholar at the Computer Laboratory of the University of Cambridge.

Authors' Addresses

Ana Minaburo
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

Laurent Toutain
IMT-Atlantique
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@imt-atlantique.fr

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

Email: carlesgo@entel.upc.edu

lpwan
Internet-Draft
Intended status: Informational
Expires: April 6, 2018

S. Farrell, Ed.
Trinity College Dublin
October 3, 2017

LPWAN Overview
draft-ietf-lpwan-overview-07

Abstract

Low Power Wide Area Networks (LPWAN) are wireless technologies with characteristics such as large coverage areas, low bandwidth, possibly very small packet and application layer data sizes and long battery life operation. This memo is an informational overview of the set of LPWAN technologies being considered in the IETF and of the gaps that exist between the needs of those technologies and the goal of running IP in LPWANs.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 6, 2018.

Copyright Notice

Copyright (c) 2017 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. LPWAN Technologies 3
 - 2.1. LoRaWAN 4
 - 2.1.1. Provenance and Documents 4
 - 2.1.2. Characteristics 4
 - 2.2. Narrowband IoT (NB-IoT) 10
 - 2.2.1. Provenance and Documents 10
 - 2.2.2. Characteristics 11
 - 2.3. SIGFOX 15
 - 2.3.1. Provenance and Documents 15
 - 2.3.2. Characteristics 15
 - 2.4. Wi-SUN Alliance Field Area Network (FAN) 20
 - 2.4.1. Provenance and Documents 20
 - 2.4.2. Characteristics 21
- 3. Generic Terminology 24
- 4. Gap Analysis 25
 - 4.1. Naive application of IPv6 26
 - 4.2. 6LOWPAN 26
 - 4.2.1. Header Compression 27
 - 4.2.2. Address Autoconfiguration 27
 - 4.2.3. Fragmentation 27
 - 4.2.4. Neighbor Discovery 28
 - 4.3. 6lo 28
 - 4.4. 6tisch 29
 - 4.5. RoHC 29
 - 4.6. ROLL 30
 - 4.7. CoAP 30
 - 4.8. Mobility 30
 - 4.9. DNS and LPWAN 31
- 5. Security Considerations 31
- 6. IANA Considerations 32
- 7. Contributors 32
- 8. Acknowledgments 34
- 9. Informative References 35
- Appendix A. Changes 40
 - A.1. From -00 to -01 40
 - A.2. From -01 to -02 40
 - A.3. From -02 to -03 41
 - A.4. From -03 to -04 41
 - A.5. From -04 to -05 41
 - A.6. From -05 to -06 41
- Author's Address 41

1. Introduction

This document provides background material and an overview of the technologies being considered in the IETF's Low Power Wide-Area Networking (LPWAN) working group. We also provide a gap analysis between the needs of these technologies and currently available IETF specifications.

Most technologies in this space aim for similar goals of supporting large numbers of very low-cost, low-throughput devices with very-low power consumption, so that even battery-powered devices can be deployed for years. LPWAN devices also tend to be constrained in their use of bandwidth, for example with limited frequencies being allowed to be used within limited duty-cycles (usually expressed as a percentage of time per-hour that the device is allowed to transmit.) And as the name implies, coverage of large areas is also a common goal. So, by and large, the different technologies aim for deployment in very similar circumstances.

Existing pilot deployments have shown huge potential and created much industrial interest in these technologies. As of today, essentially no LPWAN devices have IP capabilities. Connecting LPWANs to the Internet would provide significant benefits to these networks in terms of interoperability, application deployment, and management, among others. The goal of the IETF LPWAN working group is to, where necessary, adapt IETF-defined protocols, addressing schemes and naming to this particular constrained environment.

This document is largely the work of the people listed in Section 7.

2. LPWAN Technologies

This section provides an overview of the set of LPWAN technologies that are being considered in the LPWAN working group. The text for each was mainly contributed by proponents of each technology.

Note that this text is not intended to be normative in any sense, but simply to help the reader in finding the relevant layer 2 specifications and in understanding how those integrate with IETF-defined technologies. Similarly, there is no attempt here to set out the pros and cons of the relevant technologies.

Note that some of the technology-specific drafts referenced below may have been updated since publication of this document.

2.1. LoRaWAN

Text here is largely from [I-D.farrell-lpwan-lora-overview]

2.1.1. Provenance and Documents

LoRaWAN is an ISM-based wireless technology for long-range low-power low-data-rate applications developed by the LoRa Alliance, a membership consortium. <<https://www.lora-alliance.org/>> This draft is based on version 1.0.2 [LoRaSpec] of the LoRa specification. That specification is publicly available and has already seen several deployments across the globe.

2.1.2. Characteristics

LoRaWAN aims to support end-devices operating on a single battery for an extended period of time (e.g., 10 years or more), extended coverage through 155 dB maximum coupling loss, and reliable and efficient file download (as needed for remote software/firmware upgrade).

LoRaWAN networks are typically organized in a star-of-stars topology in which gateways relay messages between end-devices and a central "network server" in the backend. Gateways are connected to the network server via IP links while end-devices use single-hop LoRaWAN communication that can be received at one or more gateways. Communication is generally bi-directional; uplink communication from end-devices to the network server is favored in terms of overall bandwidth availability.

Figure 1 shows the entities involved in a LoRaWAN network.

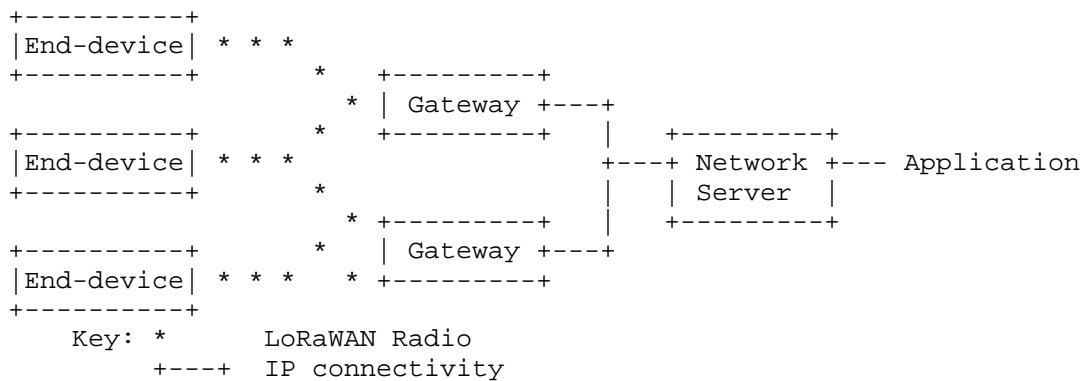


Figure 1: LoRaWAN architecture

- o End-device: a LoRa client device, sometimes called a mote. Communicates with gateways.
- o Gateway: a radio on the infrastructure-side, sometimes called a concentrator or base-station. Communicates with end-devices and, via IP, with a network server.
- o Network Server: The Network Server (NS) terminates the LoRaWAN MAC layer for the end-devices connected to the network. It is the center of the star topology.
- o - Join Server: The Join Server (JS) is a server on the Internet side of an NS that processes join requests from end-devices.
- o Uplink message: refers to communications from end-device to network server or application via one or more gateways.
- o Downlink message: refers to communications from network server or application via one gateway to a single end-device or a group of end-devices (considering multicasting).
- o Application: refers to application layer code both on the end-device and running "behind" the network server. For LoRaWAN, there will generally only be one application running on most end-devices. Interfaces between the network server and application are not further described here.

In LoRaWAN networks, end-device transmissions may be received at multiple gateways, so during nominal operation a network server may see multiple instances of the same uplink message from an end-device.

The LoRaWAN network infrastructure manages the data rate and RF output power for each end-device individually by means of an adaptive data rate (ADR) scheme. End-devices may transmit on any channel allowed by local regulation at any time.

LoRaWAN radios make use of industrial, scientific and medical (ISM) bands, for example, 433MHz and 868MHz within the European Union and 915MHz in the Americas.

The end-device changes channel in a pseudo-random fashion for every transmission to help make the system more robust to interference and/or to conform to local regulations.

Figure 2 below shows that after a transmission slot a Class A device turns on its receiver for two short receive windows that are offset from the end of the transmission window. End-devices can only transmit a subsequent uplink frame after the end of the associated

receive windows. When a device joins a LoRaWAN network, there are similar timeouts on parts of that process.

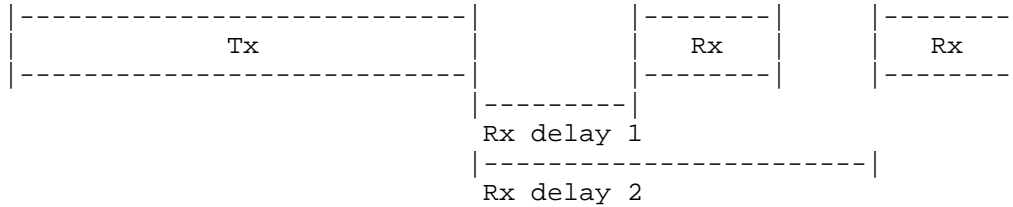


Figure 2: LoRaWAN Class A transmission and reception window

Given the different regional requirements the detailed specification for the LoRaWAN physical layer (taking up more than 30 pages of the specification) is not reproduced here. Instead and mainly to illustrate the kinds of issue encountered, in Table 1 we present some of the default settings for one ISM band (without fully explaining those here) and in Table 2 we describe maxima and minima for some parameters of interest to those defining ways to use IETF protocols over the LoRaWAN MAC layer.

Parameters	Default Value
Rx delay 1	1 s
Rx delay 2	2 s (must be RECEIVE_DELAY1 + 1s)
join delay 1	5 s
join delay 2	6 s
868MHz Default channels	3 (868.1,868.2,868.3), data rate: 0.3-5 kbps

Table 1: Default settings for EU868MHz band

Parameter/Notes	Min	Max
Duty Cycle: some but not all ISM bands impose a limit in terms of how often an end-device can transmit. In some cases LoRaWAN is more restrictive in an attempt to avoid congestion.	1%	no-limit
EU 868MHz band data rate/frame-size	250 bits/s : 59 octets	50000 bits/s : 250 octets
US 915MHz band data rate/frame-size	980 bits/s : 19 octets	21900 bits/s : 250 octets

Table 2: Minima and Maxima for various LoRaWAN Parameters

Note that in the case of the smallest frame size (19 octets), 8 octets are required for LoRa MAC layer headers leaving only 11 octets for payload (including MAC layer options). However, those settings do not apply for the join procedure - end-devices are required to use a channel and data rate that can send the 23-byte Join-request message for the join procedure.

Uplink and downlink higher layer data is carried in a MACPayload. There is a concept of "ports" (an optional 8-bit value) to handle different applications on an end-device. Port zero is reserved for LoRaWAN specific messaging, such as the configuration of the end device's network parameters (available channels, data rates, ADR parameters, RX1/2 delay, etc.).

In addition to carrying higher layer PDUs there are Join-Request and Join-Response (aka Join-Accept) messages for handling network access. And so-called "MAC commands" (see below) up to 15 bytes long can be piggybacked in an options field ("FOpts").

There are a number of MAC commands for link and device status checking, ADR and duty-cycle negotiation, managing the RX windows and radio channel settings. For example, the link check response message allows the network server (in response to a request from an end-device) to inform an end-device about the signal attenuation seen most recently at a gateway, and to also tell the end-device how many gateways received the corresponding link request MAC command.

Some MAC commands are initiated by the network server. For example, one command allows the network server to ask an end-device to reduce its duty-cycle to only use a proportion of the maximum allowed in a region. Another allows the network server to query the end-device's power status with the response from the end-device specifying whether it has an external power source or is battery powered (in which case a relative battery level is also sent to the network server).

In order to operate nominally on a LoRaWAN network, a device needs a 32-bit device address, that is assigned when the device "joins" the network (see below for the join procedure) or that is pre-provisioned into the device. In case of roaming devices, the device address is assigned based on the 24-bit network identifier (NetID) that is allocated to the network by the LoRa Alliance. Non-roaming devices can be assigned device addresses by the network without relying on a LoRa Alliance-assigned NetID.

End-devices are assumed to work with one or a quite limited number of applications, identified by a 64-bit AppEUI, which is assumed to be a registered IEEE EUI64 value. In addition, a device needs to have two symmetric session keys, one for protecting network artifacts (port=0), the NwksKey, and another for protecting application layer traffic, the AppSKey. Both keys are used for 128-bit AES cryptographic operations. So, one option is for an end-device to have all of the above, plus channel information, somehow (pre-)provisioned, in which case the end-device can simply start transmitting. This is achievable in many cases via out-of-band means given the nature of LoRaWAN networks. Table 3 summarizes these values.

Value	Description
DevAddr	DevAddr (32-bits) = device-specific network address generated from the NetID
AppEUI	IEEE EUI64 corresponding to the join server for an application
NwksKey	128-bit network session key used with AES-CMAC
AppSKey	128-bit application session key used with AES-CTR
AppKey	128-bit application session key used with AES-ECB

Table 3: Values required for nominal operation

As an alternative, end-devices can use the LoRaWAN join procedure with a join server behind the NS in order to setup some of these values and dynamically gain access to the network. To use the join procedure, an end-device must still know the AppEUI, and in addition, a different (long-term) symmetric key that is bound to the AppEUI - this is the application key (AppKey), and is distinct from the application session key (AppSKey). The AppKey is required to be specific to the device, that is, each end-device should have a different AppKey value. And finally, the end-device also needs a long-term identifier for itself, syntactically also an EUI-64, and known as the device EUI or DevEUI. Table 4 summarizes these values.

Value	Description
DevEUI	IEEE EUI64 naming the device
AppEUI	IEEE EUI64 naming the application
AppKey	128-bit long term application key for use with AES

Table 4: Values required for join procedure

The join procedure involves a special exchange where the end-device asserts the AppEUI and DevEUI (integrity protected with the long-term AppKey, but not encrypted) in a Join-request uplink message. This is then routed to the network server which interacts with an entity that knows that AppKey to verify the Join-request. All going well, a Join-accept downlink message is returned from the network server to the end-device that specifies the 24-bit NetID, 32-bit DevAddr and channel information and from which the AppSKey and NwkSKey can be derived based on knowledge of the AppKey. This provides the end-device with all the values listed in Table 3.

All payloads are encrypted and have data integrity. MAC commands, when sent as a payload (port zero), are therefore protected. MAC commands piggy-backed as frame options ("FOpts") are however sent in clear. Any MAC commands sent as frame options and not only as payload, are visible to a passive attacker but are not malleable for an active attacker due to the use of the Message Integrity Check (MIC) described below.

For LoRaWAN version 1.0.x, the NwkSKey session key is used to provide data integrity between the end-device and the network server. The AppSKey is used to provide data confidentiality between the end-device and network server, or to the application "behind" the network server, depending on the implementation of the network.

All MAC layer messages have an outer 32-bit MIC calculated using AES-CMAC calculated over the ciphertext payload and other headers and using the NwkSKey. Payloads are encrypted using AES-128, with a counter-mode derived from IEEE 802.15.4 using the AppSKey. Gateways are not expected to be provided with the AppSKey or NwkSKey, all of the infrastructure-side cryptography happens in (or "behind") the network server. When session keys are derived from the AppKey as a result of the join procedure the Join-accept message payload is specially handled.

The long-term AppKey is directly used to protect the Join-accept message content, but the function used is not an AES-encrypt operation, but rather an AES-decrypt operation. The justification is that this means that the end-device only needs to implement the AES-encrypt operation. (The counter mode variant used for payload decryption means the end-device doesn't need an AES-decrypt primitive.)

The Join-accept plaintext is always less than 16 bytes long, so electronic code book (ECB) mode is used for protecting Join-accept messages. The Join-accept contains an AppNonce (a 24 bit value) that is recovered on the end-device along with the other Join-accept content (e.g. DevAddr) using the AES-encrypt operation. Once the Join-accept payload is available to the end-device the session keys are derived from the AppKey, AppNonce and other values, again using an ECB mode AES-encrypt operation, with the plaintext input being a maximum of 16 octets.

2.2. Narrowband IoT (NB-IoT)

Text here is largely from [I-D.ratilainen-lpwan-nb-iot]

2.2.1. Provenance and Documents

Narrowband Internet of Things (NB-IoT) is developed and standardized by 3GPP. The standardization of NB-IoT was finalized with 3GPP Release 13 in June 2016, and further enhancements for NB-IoT are specified in 3GPP Release 14 in 2017, for example in the form of multicast support. Further features and improvements will be developed in the following releases, but NB-IoT has been ready to be deployed since 2016, and is rather simple to deploy especially in the existing LTE networks with a software upgrade in the operator's base stations. For more information of what has been specified for NB-IoT, 3GPP specification 36.300 [TGPP36300] provides an overview and overall description of the E-UTRAN radio interface protocol architecture, while specifications 36.321 [TGPP36321], 36.322 [TGPP36322], 36.323 [TGPP36323] and 36.331 [TGPP36331] give more detailed description of MAC, RLC, PDCP and RRC protocol layers,

respectively. Note that the description below assumes familiarity with numerous 3GPP terms.

2.2.2. Characteristics

Specific targets for NB-IoT include: Less than US\$5 module cost, extended coverage of 164 dB maximum coupling loss, battery life of over 10 years, ~55000 devices per cell and uplink reporting latency of less than 10 seconds.

NB-IoT supports Half Duplex FDD operation mode with 60 kbps peak rate in uplink and 30 kbps peak rate in downlink, and a maximum transmission unit (MTU) size of 1600 bytes limited by PDCP layer (see Figure 4 for the protocol structure), which is the highest layer in the user plane, as explained later. Any packet size up to the said MTU size can be passed to the NB-IoT stack from higher layers, segmentation of the packet is performed in the RLC layer, which can segment the data to transmission blocks with size as small as 16 bits. As the name suggests, NB-IoT uses narrowbands with bandwidth of 180 kHz in both downlink and uplink. The multiple access scheme used in the downlink is OFDMA with 15 kHz sub-carrier spacing. In uplink, SC-FDMA single tone with either 15kHz or 3.75 kHz tone spacing is used, or optionally multi-tone SC-FDMA can be used with 15 kHz tone spacing.

NB-IoT can be deployed in three ways. In-band deployment means that the narrowband is deployed inside the LTE band and radio resources are flexibly shared between NB-IoT and normal LTE carrier. In Guard-band deployment the narrowband uses the unused resource blocks between two adjacent LTE carriers. Standalone deployment is also supported, where the narrowband can be located alone in dedicated spectrum, which makes it possible for example to reframe a GSM carrier at 850/900 MHz for NB-IoT. All three deployment modes are used in licensed frequency bands. The maximum transmission power is either 20 or 23 dBm for uplink transmissions, while for downlink transmission the eNodeB may use higher transmission power, up to 46 dBm depending on the deployment.

A maximum coupling loss (MCL) target for NB-IoT coverage enhancements defined by 3GPP is 164 dB. With this MCL, the performance of NB-IoT in downlink varies between 200 bps and 2-3 kbps, depending on the deployment mode. Stand-alone operation may achieve the highest data rates, up to few kbps, while in-band and guard-band operations may reach several hundreds of bps. NB-IoT may even operate with MCL higher than 170 dB with very low bit rates.

For signaling optimization, two options are introduced in addition to legacy LTE RRC connection setup; mandatory Data-over-NAS (Control

Plane optimization, solution 2 in [TGPP23720]) and optional RRC Suspend/Resume (User Plane optimization, solution 18 in [TGPP23720]). In the control plane optimization the data is sent over Non-Access Stratum, directly to/from Mobility Management Entity (MME) (see Figure 3 for the network architecture) in the core network to the User Equipment (UE) without interaction from the base station. This means there are no Access Stratum security or header compression provided by the PDCP layer in the eNodeB, as the Access Stratum is bypassed, and only limited RRC procedures. RoHC based header compression may still optionally be provided and terminated in MME.

The RRC Suspend/Resume procedures reduce the signaling overhead required for UE state transition from RRC Idle to RRC Connected mode compared to legacy LTE operation in order to have quicker user plane transaction with the network and return to RRC Idle mode faster.

In order to prolong device battery life, both power-saving mode (PSM) and extended DRX (eDRX) are available to NB-IoT. With eDRX the RRC Connected mode DRX cycle is up to 10.24 seconds and in RRC Idle the eDRX cycle can be up to 3 hours. In PSM the device is in a deep sleep state and only wakes up for uplink reporting, after which there is a window, configured by the network, during which the device receiver is open for downlink connectivity, of for periodical "keep-alive" signaling (PSM uses periodic TAU signaling with additional reception window for downlink reachability).

Since NB-IoT operates in licensed spectrum, it has no channel access restrictions allowing up to a 100% duty-cycle.

3GPP access security is specified in [TGPP33203].

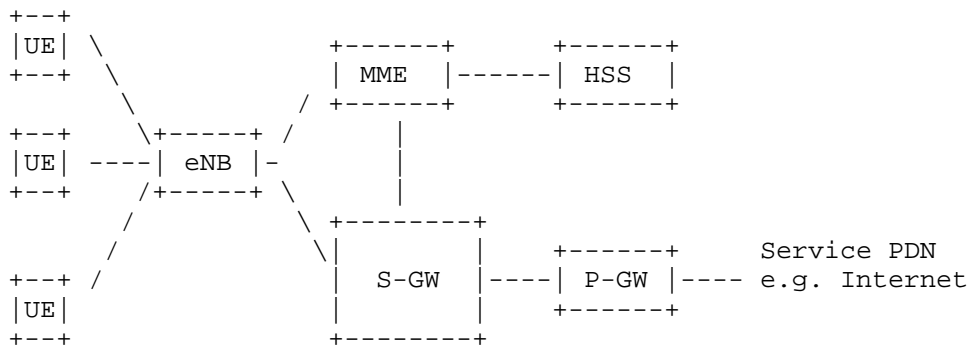


Figure 3: 3GPP network architecture

Figure 3 shows the 3GPP network architecture, which applies to NB-IoT. Mobility Management Entity (MME) is responsible for handling

the mobility of the UE. MME tasks include tracking and paging UEs, session management, choosing the Serving gateway for the UE during initial attachment and authenticating the user. At MME, the Non-Access Stratum (NAS) signaling from the UE is terminated.

Serving Gateway (S-GW) routes and forwards the user data packets through the access network and acts as a mobility anchor for UEs during handover between base stations known as eNodeBs and also during handovers between NB-IoT and other 3GPP technologies.

Packet Data Network Gateway (P-GW) works as an interface between 3GPP network and external networks.

The Home Subscriber Server (HSS) contains user-related and subscription-related information. It is a database, which performs mobility management, session establishment support, user authentication and access authorization.

E-UTRAN consists of components of a single type, eNodeB. eNodeB is a base station, which controls the UEs in one or several cells.

The 3GPP radio protocol architecture is illustrated in Figure 4.

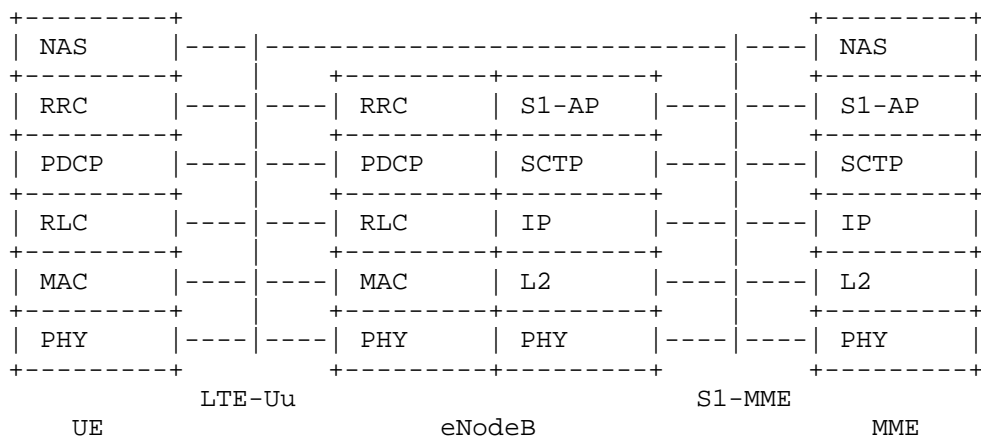


Figure 4: 3GPP radio protocol architecture for control plane

Control plane protocol stack

The radio protocol architecture of NB-IoT (and LTE) is separated into control plane and user plane. The control plane consists of protocols which control the radio access bearers and the connection between the UE and the network. The highest layer of control plane is called Non-Access Stratum (NAS), which conveys the radio signaling

between the UE and the Evolved Packet Core (EPC), passing transparently through the radio network. NAS responsible for authentication, security control, mobility management and bearer management.

Access Stratum (AS) is the functional layer below NAS, and in the control plane it consists of Radio Resource Control protocol (RRC) [TGPP36331], which handles connection establishment and release functions, broadcast of system information, radio bearer establishment, reconfiguration and release. RRC configures the user and control planes according to the network status. There exists two RRC states, RRC_Idle or RRC_Connected, and RRC entity controls the switching between these states. In RRC_Idle, the network knows that the UE is present in the network and the UE can be reached in case of incoming call/downlink data. In this state, the UE monitors paging, performs cell measurements and cell selection and acquires system information. Also the UE can receive broadcast and multicast data, but it is not expected to transmit or receive unicast data. In RRC_Connected the UE has a connection to the eNodeB, the network knows the UE location on the cell level and the UE may receive and transmit unicast data. An RRC connection is established when the UE is expected to be active in the network, to transmit or receive data. The RRC connection is released, switching back to RRC_Idle, when there is no more traffic in order to preserve UE battery life and radio resources. However, a new feature was introduced for NB-IoT, as mentioned earlier, which allows data to be transmitted from the MME directly to the UE transparently to the eNodeB, thus bypassing AS functions.

Packet Data Convergence Protocol's (PDCP) [TGPP36323] main services in control plane are transfer of control plane data, ciphering and integrity protection.

Radio Link Control protocol (RLC) [TGPP36322] performs transfer of upper layer PDUs and optionally error correction with Automatic Repeat reQuest (ARQ), concatenation, segmentation, and reassembly of RLC SDUs, in-sequence delivery of upper layer PDUs, duplicate detection, RLC SDU discard, RLC-re-establishment and protocol error detection and recovery.

Medium Access Control protocol (MAC) [TGPP36321] provides mapping between logical channels and transport channels, multiplexing of MAC SDUs, scheduling information reporting, error correction with HARQ, priority handling and transport format selection.

Physical layer [TGPP36201] provides data transport services to higher layers. These include error detection and indication to higher layers, FEC encoding, HARQ soft-combining, rate matching and mapping

of the transport channels onto physical channels, power weighting and modulation of physical channels, frequency and time synchronization and radio characteristics measurements.

User plane protocol stack

User plane is responsible for transferring the user data through the Access Stratum. It interfaces with IP and the highest layer of user plane is PDCP, which in user plane performs header compression using Robust Header Compression (RoHC), transfer of user plane data between eNodeB and UE, ciphering and integrity protection. Similar to control plane, lower layers in user plane include RLC, MAC and physical layer performing the same tasks as in control plane.

2.3. SIGFOX

Text here is largely from [I-D.zuniga-lpwan-sigfox-system-description] which may have been updated since this was published.

2.3.1. Provenance and Documents

The SIGFOX LPWAN is in line with the terminology and specifications being defined by ETSI [etsi_unb]. As of today, SIGFOX's network has been fully deployed in 12 countries, with ongoing deployments on 26 other countries, giving in total a geography of 2 million square kilometers, containing 512 million people.

2.3.2. Characteristics

SIGFOX LPWAN autonomous battery-operated devices send only a few bytes per day, week or month, in principle allowing them to remain on a single battery for up to 10-15 years. Hence, the system is designed as to allow devices to last several years, sometimes even buried underground.

Since the radio protocol is connection-less and optimized for uplink communications, the capacity of a SIGFOX base station depends on the number of messages generated by devices, and not on the actual number of devices. Likewise, the battery life of devices depends on the number of messages generated by the device. Depending on the use case, devices can vary from sending less than one message per device per day, to dozens of messages per device per day.

The coverage of the cell depends on the link budget and on the type of deployment (urban, rural, etc.). The radio interface is compliant with the following regulations:

Spectrum allocation in the USA [fcc_ref]

Spectrum allocation in Europe [etsi_ref]

Spectrum allocation in Japan [arib_ref]

The SIGFOX radio interface is also compliant with the local regulations of the following countries: Australia, Brazil, Canada, Kenya, Lebanon, Mauritius, Mexico, New Zealand, Oman, Peru, Singapore, South Africa, South Korea, and Thailand.

The radio interface is based on Ultra Narrow Band (UNB) communications, which allow an increased transmission range by spending a limited amount of energy at the device. Moreover, UNB allows a large number of devices to coexist in a given cell without significantly increasing the spectrum interference.

Both uplink and downlink are supported, although the system is optimized for uplink communications. Due to spectrum optimizations, different uplink and downlink frames and time synchronization methods are needed.

The main radio characteristics of the UNB uplink transmission are:

- o Channelization mask: 100 Hz / 600 Hz (depending on the region)
- o Uplink baud rate: 100 baud / 600 baud (depending on the region)
- o Modulation scheme: DBPSK
- o Uplink transmission power: compliant with local regulation
- o Link budget: 155 dB (or better)
- o Central frequency accuracy: not relevant, provided there is no significant frequency drift within an uplink packet transmission

For example, in Europe the UNB uplink frequency band is limited to 868.00 to 868.60 MHz, with a maximum output power of 25 mW and a duty cycle of 1%.

The format of the uplink frame is the following:

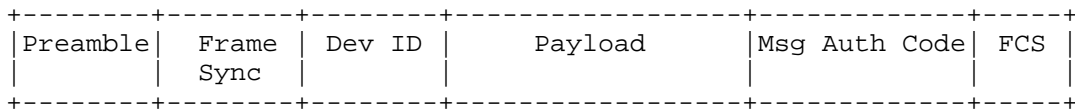


Figure 5: Uplink Frame Format

The uplink frame is composed of the following fields:

- o Preamble: 19 bits
- o Frame sync and header: 29 bits
- o Device ID: 32 bits
- o Payload: 0-96 bits
- o Authentication: 16-40 bits
- o Frame check sequence: 16 bits (CRC)

The main radio characteristics of the UNB downlink transmission are:

- o Channelization mask: 1.5 kHz
- o Downlink baud rate: 600 baud
- o Modulation scheme: GFSK
- o Downlink transmission power: 500 mW / 4W (depending on the region)
- o Link budget: 153 dB (or better)
- o Central frequency accuracy: the center frequency of downlink transmission is set by the network according to the corresponding uplink transmission

For example, in Europe the UNB downlink frequency band is limited to 869.40 to 869.65 MHz, with a maximum output power of 500 mW with 10% duty cycle.

The format of the downlink frame is the following:

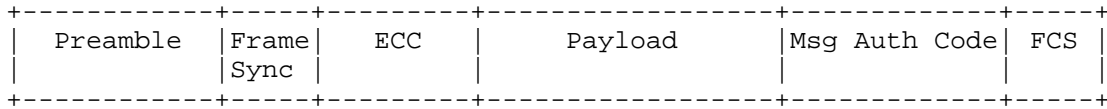


Figure 6: Downlink Frame Format

The downlink frame is composed of the following fields:

- o Preamble: 91 bits
- o Frame sync and header: 13 bits
- o Error Correcting Code (ECC): 32 bits
- o Payload: 0-64 bits
- o Authentication: 16 bits
- o Frame check sequence: 8 bits (CRC)

The radio interface is optimized for uplink transmissions, which are asynchronous. Downlink communications are achieved by devices querying the network for available data.

A device willing to receive downlink messages opens a fixed window for reception after sending an uplink transmission. The delay and duration of this window have fixed values. The network transmits the downlink message for a given device during the reception window, and the network also selects the base station (BS) for transmitting the corresponding downlink message.

Uplink and downlink transmissions are unbalanced due to the regulatory constraints on ISM bands. Under the strictest regulations, the system can allow a maximum of 140 uplink messages and 4 downlink messages per device per day. These restrictions can be slightly relaxed depending on system conditions and the specific regulatory domain of operation.

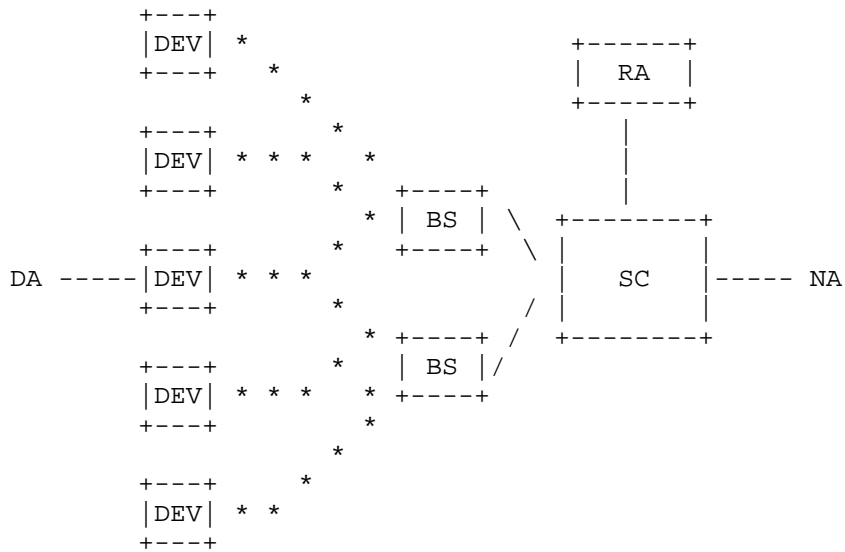


Figure 7: SIGFOX network architecture

Figure 7 depicts the different elements of the SIGFOX network architecture.

SIGFOX has a "one-contract one-network" model allowing devices to connect in any country, without any need or notion of either roaming or handover.

The architecture consists of a single cloud-based core network, which allows global connectivity with minimal impact on the end device and radio access network. The core network elements are the Service Center (SC) and the Registration Authority (RA). The SC is in charge of the data connectivity between the Base Station (BS) and the Internet, as well as the control and management of the BSs and End Points. The RA is in charge of the End Point network access authorization.

The radio access network is comprised of several BSs connected directly to the SC. Each BS performs complex L1/L2 functions, leaving some L2 and L3 functionalities to the SC.

The Devices (DEVs) or End Points (EPs) are the objects that communicate application data between local device applications (DAs) and network applications (NAs).

Devices (or EPs) can be static or nomadic, as they associate with the SC and they do not attach to any specific BS. Hence, they can communicate with the SC through one or multiple BSs.

Due to constraints in the complexity of the Device, it is assumed that Devices host only one or very few device applications, which most of the time communicate each to a single network application at a time.

The radio protocol authenticates and ensures the integrity of each message. This is achieved by using a unique device ID and an AES-128 based message authentication code, ensuring that the message has been generated and sent by the device with the ID claimed in the message. Application data can be encrypted at the application level or not, depending on the criticality of the use case, to provide a balance between cost and effort vs. risk. AES-128 in counter mode is used for encryption. Cryptographic keys are independent for each device. These keys are associated with the device ID and separate integrity and confidentiality keys are pre-provisioned. A confidentiality key is only provisioned if confidentiality is to be used. At the time of writing the algorithms and keying details for this are not published.

2.4. Wi-SUN Alliance Field Area Network (FAN)

Text here is via personal communication from Bob Heile (bheile@ieee.org) and was authored by Bob and Sum Chin Sean. Duffy (paduffy@cisco.com) also provided additional comments/input on this section.

2.4.1. Provenance and Documents

The Wi-SUN Alliance <<https://www.wi-sun.org/>> is an industry alliance for smart city, smart grid, smart utility, and a broad set of general IoT applications. The Wi-SUN Alliance Field Area Network (FAN) profile is open standards based (primarily on IETF and IEEE802 standards) and was developed to address applications like smart municipality/city infrastructure monitoring and management, electric vehicle (EV) infrastructure, advanced metering infrastructure (AMI), distribution automation (DA), supervisory control and data acquisition (SCADA) protection/management, distributed generation monitoring and management, and many more IoT applications. Additionally, the Alliance has created a certification program to promote global multi-vendor interoperability.

The FAN profile is specified within ANSI/TIA as an extension of work previously done on Smart Utility Networks. [ANSI-4957-000]. Updates to those specifications intended to be published in 2017 will contain details of the FAN profile. A current snapshot of the work to

produce that profile is presented in [wisun-pressie1]
[wisun-pressie2] .

2.4.2. Characteristics

The FAN profile is an IPv6 wireless mesh network with support for enterprise level security. The frequency hopping wireless mesh topology aims to offer superior network robustness, reliability due to high redundancy, good scalability due to the flexible mesh configuration and good resilience to interference. Very low power modes are in development permitting long term battery operation of network nodes.

The following list contains some overall characteristics of Wi-SUN that are relevant to LPWAN applications.

- o Coverage The range of Wi-SUN FAN is typically 2 -- 3 km in line of sight, matching the needs of neighborhood area networks, campus area networks, or corporate area networks. The range can also be extended via multi-hop networking.
- o High bandwidth, low link latency: Wi-SUN supports relatively high bandwidth, i.e. up to 300 kbps [FANTPS], enables remote update and upgrade of devices so that they can handle new applications, extending their working life. Wi-SUN supports LPWAN IoT applications that require on-demand control by providing low link latency (0.02s) and bi-directional communication.
- o Low power consumption: FAN devices draw less than 2 uA when resting and only 8 mA when listening. Such devices can maintain a long lifetime even if they are frequently listening. For instance, suppose the device transmits data for 10 ms once every 10 s; theoretically, a battery of 1000 mAh can last more than 10 years.
- o Scalability: Tens of millions Wi-SUN FAN devices have been deployed in urban, suburban and rural environments, including deployments with more than 1 million devices.

A FAN contains one or more networks. Within a network, nodes assume one of three operational roles. First, each network contains a Border Router providing Wide Area Network (WAN) connectivity to the network. The Border Router maintains source routing tables for all nodes within its network, provides node authentication and key management services, and disseminates network-wide information such as broadcast schedules. Secondly, Router nodes, which provide upward and downward packet forwarding (within a network). A Router also provides services for relaying security and address management

protocols. Lastly, Leaf nodes provide minimum capabilities: discovering and joining a network, send/receive IPv6 packets, etc. A low power network may contain a mesh topology with Routers at the edges that construct a star topology with Leaf nodes.

The FAN profile is based on various open standards developed by the IETF (including [RFC0768], [RFC2460], [RFC4443] and [RFC6282]), IEEE802 (including [IEEE-802-15-4] and [IEEE-802-15-9]) and ANSI/TIA [ANSI-4957-210] for low power and lossy networks.

The FAN profile specification provides an application-independent IPv6-based transport service. There are two possible methods for establishing the IPv6 packet routing: Routing Protocol for Low-Power and Lossy Networks (RPL) at the Network layer is mandatory, and Multi-Hop Delivery Service (MHDS) is optional at the Data Link layer. Table 5 provides an overview of the FAN network stack.

The Transport service is based on User Datagram Protocol (UDP) defined in RFC768 or Transmission Control Protocol (TCP) defined in RFC793.

The Network service is provided by IPv6 as defined in RFC2460 with 6LoWPAN adaptation as defined in RFC4944 and RFC6282. ICMPv6, as defined in RFC4443, is used for the control plane during information exchange.

The Data Link service provides both control/management of the Physical layer and data transfer/management services to the Network layer. These services are divided into Media Access Control (MAC) and Logical Link Control (LLC) sub-layers. The LLC sub-layer provides a protocol dispatch service which supports 6LoWPAN and an optional MAC sub-layer mesh service. The MAC sub-layer is constructed using data structures defined in IEEE802.15.4-2015. Multiple modes of frequency hopping are defined. The entire MAC payload is encapsulated in an IEEE802.15.9 Information Element to enable LLC protocol dispatch between upper layer 6LoWPAN processing, MAC sublayer mesh processing, etc. These areas will be expanded once IEEE802.15.12 is completed.

The PHY service is derived from a sub-set of the SUN FSK specification in IEEE802.15.4-2015. The 2-FSK modulation schemes, with channel spacing range from 200 to 600 kHz, are defined to provide data rates from 50 to 300 kbps, with Forward Error Coding (FEC) as an optional feature. Towards enabling ultra-low-power applications, the PHY layer design is also extendable to low energy and critical infrastructure monitoring networks.

Layer	Description
IPv6 protocol suite	TCP/UDP 6LoWPAN Adaptation + Header Compression DHCPv6 for IP address management. Routing using RPL. ICMPv6. Unicast and Multicast forwarding.
MAC based on IEEE 802.15.4e + IE extensions	Frequency hopping Discovery and Join Protocol Dispatch (IEEE 802.15.9) Several Frame Exchange patterns Optional Mesh Under routing (ANSI 4957.210).
PHY based on 802.15.4g	Various data rates and regions
Security	802.1X/EAP-TLS/PKI Authentication. TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 required for EAP-TLS. 802.11i Group Key Management Frame security is implemented as AES-CCM* as specified in IEEE 802.15.4 Optional ETSI-TS-102-887-2 Node 2 Node Key Management

Table 5: Wi-SUN Stack Overview

The FAN security supports Data Link layer network access control, mutual authentication, and establishment of a secure pairwise link

between a FAN node and its Border Router, which is implemented with an adaptation of IEEE802.1X and EAP-TLS as described in [RFC5216] using secure device identity as described in IEEE802.1AR. Certificate formats are based upon [RFC5280]. A secure group link between a Border Router and a set of FAN nodes is established using an adaptation of the IEEE802.11 Four-Way Handshake. A set of 4 group keys are maintained within the network, one of which is the current transmit key. Secure node to node links are supported between one-hop FAN neighbors using an adaptation of ETSI-TS-102-887-2. FAN nodes implement Frame Security as specified in IEEE802.15.4-2015.

3. Generic Terminology

LPWAN technologies, such as those discussed above, have similar architectures but different terminology. We can identify different types of entities in a typical LPWAN network:

- o End-Devices are the devices or the "things" (e.g. sensors, actuators, etc.); they are named differently in each technology (End Device, User Equipment or End Point). There can be a high density of end devices per radio gateway.
- o The Radio Gateway, which is the end point of the constrained link. It is known as: Gateway, Evolved Node B or Base station.
- o The Network Gateway or Router is the interconnection node between the Radio Gateway and the Internet. It is known as: Network Server, Serving GW or Service Center.
- o LPWAN-AAA Server, which controls the user authentication, the applications. It is known as: Join-Server, Home Subscriber Server or Registration Authority. (We use the term LPWAN-AAA server because we're not assuming that this entity speaks RADIUS or Diameter as many/most AAA servers do, but equally we don't want to rule that out, as the functionality will be similar.
- o At last we have the Application Server, known also as Packet Data Node Gateway or Network Application.

Function/ Technology	LORAWAN	NB-IOT	SIGFOX	Wi-SUN	IETF
Sensor, Actuator, device, object	End Device	User Equipment	End Point	Leaf Node	Device (Dev)
Transceiver Antenna	Gateway	Evolved Node B	Base Station	Router Node	RADIO Gateway
Server	Network Server	PDN GW/ SCEF	Service Center	Border Router	Network Gateway (NGW)
Security Server	Join Server	Home Subscriber Server	Registration Authority	Authent. Server	LPWAN- AAA SERVER
Application	Application Server	Application Server	Network Application	Appli- cation	Application (App)

Figure 8: LPWAN Architecture Terminology

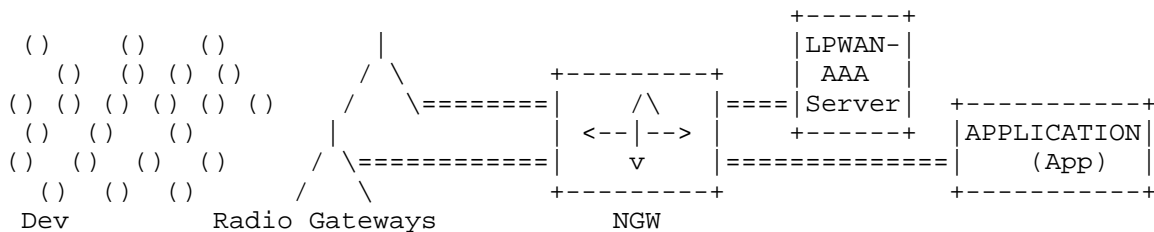


Figure 9: LPWAN Architecture

In addition to the names of entities, LPWANs are also subject to possibly regional frequency band regulations. Those may include restrictions on the duty-cycle, for example requiring that hosts only transmit for a certain percentage of each hour.

4. Gap Analysis

4.1. Naive application of IPv6

IPv6 [RFC2460] has been designed to allocate addresses to all the nodes connected to the Internet. Nevertheless, the header overhead of at least 40 bytes introduced by the protocol is incompatible with LPWAN constraints. If IPv6 with no further optimization were used, several LPWAN frames could be needed just to carry the IP header. Another problem arises from IPv6 MTU requirements, which require the layer below to support at least 1280 byte packets [RFC2460].

IPv6 has a configuration protocol - neighbor discovery protocol, (NDP) [RFC4861]). For a node to learn network parameters NDP generates regular traffic with a relatively large message size that does not fit LPWAN constraints.

In some LPWAN technologies, layer two multicast is not supported. In that case, if the network topology is a star, the solution and considerations of section 3.2.5 of [RFC7668] may be applied.

Other key protocols such as DHCPv6 [RFC3315], IPsec [RFC4301] and TLS [RFC5246] have similarly problematic properties in this context. Each of those require relatively frequent round-trips between the host and some other host on the network. In the case of cryptographic protocols such as IPsec and TLS, in addition to the round-trips required for secure session establishment, cryptographic operations can require padding and addition of authenticators that are problematic when considering LPWAN lower layers. Note that mains powered Wi-SUN mesh router nodes will typically be more resource capable than the other LPWAN techs discussed. This can enable use of more "chatty" protocols for some aspects of Wi-SUN.

4.2. 6LoWPAN

Several technologies that exhibit significant constraints in various dimensions have exploited the 6LoWPAN suite of specifications [RFC4944], [RFC6282], [RFC6775] to support IPv6 [I-D.hong-6lo-use-cases]. However, the constraints of LPWANs, often more extreme than those typical of technologies that have (re)used 6LoWPAN, constitute a challenge for the 6LoWPAN suite in order to enable IPv6 over LPWAN. LPWANs are characterized by device constraints (in terms of processing capacity, memory, and energy availability), and specially, link constraints, such as:

- o very low layer two payload size (from ~10 to ~100 bytes),
- o very low bit rate (from ~10 bit/s to ~100 kbit/s), and

- o in some specific technologies, further message rate constraints (e.g. between ~0.1 message/minute and ~1 message/minute) due to regional regulations that limit the duty cycle.

4.2.1. Header Compression

6LoWPAN header compression reduces IPv6 (and UDP) header overhead by eliding header fields when they can be derived from the link layer, and by assuming that some of the header fields will frequently carry expected values. 6LoWPAN provides both stateless and stateful header compression. In the latter, all nodes of a 6LoWPAN are assumed to share compression context. In the best case, the IPv6 header for link-local communication can be reduced to only 2 bytes. For global communication, the IPv6 header may be compressed down to 3 bytes in the most extreme case. However, in more practical situations, the smallest IPv6 header size may be 11 bytes (one address prefix compressed) or 19 bytes (both source and destination prefixes compressed). These headers are large considering the link layer payload size of LPWAN technologies, and in some cases are even bigger than the LPWAN PDUs. 6LoWPAN has been initially designed for IEEE 802.15.4 networks with a frame size up to 127 bytes and a throughput of up to 250 kb/s, which may or may not be duty-cycled.

4.2.2. Address Autoconfiguration

Traditionally, Interface Identifiers (IIDs) have been derived from link layer identifiers [RFC4944]. This allows optimizations such as header compression. Nevertheless, recent guidance has given advice on the fact that, due to privacy concerns, 6LoWPAN devices should not be configured to embed their link layer addresses in the IID by default.

4.2.3. Fragmentation

As stated above, IPv6 requires the layer below to support an MTU of 1280 bytes [RFC2460]. Therefore, given the low maximum payload size of LPWAN technologies, fragmentation is needed.

If a layer of an LPWAN technology supports fragmentation, proper analysis has to be carried out to decide whether the fragmentation functionality provided by the lower layer or fragmentation at the adaptation layer should be used. Otherwise, fragmentation functionality shall be used at the adaptation layer.

6LoWPAN defined a fragmentation mechanism and a fragmentation header to support the transmission of IPv6 packets over IEEE 802.15.4 networks [RFC4944]. While the 6LoWPAN fragmentation header is appropriate for IEEE 802.15.4-2003 (which has a frame payload size of

81-102 bytes), it is not suitable for several LPWAN technologies, many of which have a maximum payload size that is one order of magnitude below that of IEEE 802.15.4-2003. The overhead of the 6LoWPAN fragmentation header is high, considering the reduced payload size of LPWAN technologies and the limited energy availability of the devices using such technologies. Furthermore, its datagram offset field is expressed in increments of eight octets. In some LPWAN technologies, the 6LoWPAN fragmentation header plus eight octets from the original datagram exceeds the available space in the layer two payload. In addition, the MTU in the LPWAN networks could be variable which implies a variable fragmentation solution.

4.2.4. Neighbor Discovery

6LoWPAN Neighbor Discovery [RFC6775] defined optimizations to IPv6 Neighbor Discovery [RFC4861], in order to adapt functionality of the latter for networks of devices using IEEE 802.15.4 or similar technologies. The optimizations comprise host-initiated interactions to allow for sleeping hosts, replacement of multicast-based address resolution for hosts by an address registration mechanism, multihop extensions for prefix distribution and duplicate address detection (note that these are not needed in a star topology network), and support for 6LoWPAN header compression.

6LoWPAN Neighbor Discovery may be used in not so severely constrained LPWAN networks. The relative overhead incurred will depend on the LPWAN technology used (and on its configuration, if appropriate). In certain LPWAN setups (with a maximum payload size above ~60 bytes, and duty-cycle-free or equivalent operation), an RS/RA/NS/NA exchange may be completed in a few seconds, without incurring packet fragmentation.

In other LPWANs (with a maximum payload size of ~10 bytes, and a message rate of ~0.1 message/minute), the same exchange may take hours or even days, leading to severe fragmentation and consuming a significant amount of the available network resources. 6LoWPAN Neighbor Discovery behavior may be tuned through the use of appropriate values for the default Router Lifetime, the Valid Lifetime in the PIOs, and the Valid Lifetime in the 6LowPan Context Option (6CO), as well as the address Registration Lifetime. However, for the latter LPWANs mentioned above, 6LoWPAN Neighbor Discovery is not suitable.

4.3. 6lo

The 6lo WG has been reusing and adapting 6LoWPAN to enable IPv6 support over link layer technologies such as Bluetooth Low Energy (BTLE), ITU-T G.9959, DECT-ULE, MS/TP-RS485, NFC IEEE 802.11ah. (See

<<https://tools.ietf.org/wg/6lo>> for details.) These technologies are similar in several aspects to IEEE 802.15.4, which was the original 6LoWPAN target technology.

6lo has mostly used the subset of 6LoWPAN techniques best suited for each lower layer technology, and has provided additional optimizations for technologies where the star topology is used, such as BTLE or DECT-ULE.

The main constraint in these networks comes from the nature of the devices (constrained devices), whereas in LPWANs it is the network itself that imposes the most stringent constraints.

4.4. 6tisch

The 6tisch solution is dedicated to mesh networks that operate using 802.15.4e MAC with a deterministic slotted channel. The time slot channel (TSCH) can help to reduce collisions and to enable a better balance over the channels. It improves the battery life by avoiding the idle listening time for the return channel.

A key element of 6tisch is the use of synchronization to enable determinism. TSCH and 6TiSCH may provide a standard scheduling function. The LPWAN networks probably will not support synchronization like the one used in 6tisch.

4.5. RoHC

Robust header compression (RoHC) is a header compression mechanism [RFC3095] developed for multimedia flows in a point to point channel. RoHC uses 3 levels of compression, each level having its own header format. In the first level, RoHC sends 52 bytes of header, in the second level the header could be from 34 to 15 bytes and in the third level header size could be from 7 to 2 bytes. The level of compression is managed by a sequence number, which varies in size from 2 bytes to 4 bits in the minimal compression. SN compression is done with an algorithm called W-LSB (Window- Least Significant Bits). This window has a 4-bit size representing 15 packets, so every 15 packets RoHC needs to slide the window in order to receive the correct sequence number, and sliding the window implies a reduction of the level of compression. When packets are lost or errored, the decompressor loses context and drops packets until a bigger header is sent with more complete information. To estimate the performance of RoHC, an average header size is used. This average depends on the transmission conditions, but most of the time is between 3 and 4 bytes.

RoHC has not been adapted specifically to the constrained hosts and networks of LPWANs: it does not take into account energy limitations nor the transmission rate, and RoHC context is synchronised during transmission, which does not allow better compression.

4.6. ROLL

Most technologies considered by the lpwan WG are based on a star topology, which eliminates the need for routing at that layer. Future work may address additional use-cases that may require adaptation of existing routing protocols or the definition of new ones. As of the time of writing, work similar to that done in the ROLL WG and other routing protocols are out of scope of the LPWAN WG.

4.7. CoAP

CoAP [RFC7252] provides a RESTful framework for applications intended to run on constrained IP networks. It may be necessary to adapt CoAP or related protocols to take into account for the extreme duty cycles and the potentially extremely limited throughput of LPWANs.

For example, some of the timers in CoAP may need to be redefined. Taking into account CoAP acknowledgments may allow the reduction of L2 acknowledgments. On the other hand, the current work in progress in the CoRE WG where the COMI/CoOL network management interface which, uses Structured Identifiers (SID) to reduce payload size over CoAP may prove to be a good solution for the LPWAN technologies. The overhead is reduced by adding a dictionary which matches a URI to a small identifier and a compact mapping of the YANG model into the CBOR binary representation.

4.8. Mobility

LPWAN nodes can be mobile. However, LPWAN mobility is different from the one specified for Mobile IP. LPWAN implies sporadic traffic and will rarely be used for high-frequency, real-time communications. The applications do not generate a flow, they need to save energy and most of the time the node will be down.

In addition, LPWAN mobility may mostly apply to groups of devices, that represent a network in which case mobility is more a concern for the gateway than the devices. NEMO [RFC3963] Mobility or other mobile gateway solutions (such as a gateway with an LTE uplink) may be used in the case where some end-devices belonging to the same network gateway move from one point to another such that they are not aware of being mobile.

4.9. DNS and LPWAN

The Domain Name System (DNS) [RFC1035], enables applications to name things with a globally resolvable name. Many protocols use the DNS to identify hosts, for example applications using CoAP.

The DNS query/answer protocol as a pre-cursor to other communication within the time-to-live (TTL) of a DNS answer is clearly problematic in an LPWAN, say where only one round-trip per hour can be used, and with a TTL that is less than 3600. It is currently unclear whether and how DNS-like functionality might be provided in LPWANs.

5. Security Considerations

Most LPWAN technologies integrate some authentication or encryption mechanisms that were defined outside the IETF. The working group may need to do work to integrate these mechanisms to unify management. A standardized Authentication, Accounting, and Authorization (AAA) infrastructure [RFC2904] may offer a scalable solution for some of the security and management issues for LPWANs. AAA offers centralized management that may be of use in LPWANs, for example [I-D.garcia-dime-diameter-lorawan] and [I-D.garcia-radext-radius-lorawan] suggest possible security processes for a LoRaWAN network. Similar mechanisms may be useful to explore for other LPWAN technologies.

Some applications using LPWANs may raise few or no privacy considerations. For example, temperature sensors in a large office building may not raise privacy issues. However, the same sensors, if deployed in a home environment and especially if triggered due to human presence, can raise significant privacy issues - if an end-device emits (an encrypted) packet every time someone enters a room in a home, then that traffic is privacy sensitive. And the more that the existence of that traffic is visible to network entities, the more privacy sensitivities arise. At this point, it is not clear whether there are workable mitigations for problems like this - in a more typical network, one would consider defining padding mechanisms and allowing for cover traffic. In some LPWANs, those mechanisms may not be feasible. Nonetheless, the privacy challenges do exist and can be real and so some solutions will be needed. Note that many aspects of solutions in this space may not be visible in IETF specifications, but can be e.g. implementation or deployment specific.

Another challenge for LPWANs will be how to handle key management and associated protocols. In a more traditional network (e.g. the web), servers can "staple" Online Certificate Status Protocol (OCSP) responses in order to allow browsers to check revocation status for

presented certificates. [RFC6961] While the stapling approach is likely something that would help in an LPWAN, as it avoids an RTT, certificates and OCSP responses are bulky items and will prove challenging to handle in LPWANs with bounded bandwidth.

6. IANA Considerations

There are no IANA considerations related to this memo.

7. Contributors

As stated above this document is mainly a collection of content developed by the full set of contributors listed below. The main input documents and their authors were:

- o Text for Section 2.1 was provided by Alper Yegin and Stephen Farrell in [I-D.farrell-lpwan-lora-overview].
- o Text for Section 2.2 was provided by Antti Ratilainen in [I-D.ratilainen-lpwan-nb-iot].
- o Text for Section 2.3 was provided by Juan Carlos Zuniga and Benoit Ponsard in [I-D.zuniga-lpwan-sigfox-system-description].
- o Text for Section 2.4 was provided via personal communication from Bob Heile (bheile@ieee.org) and was authored by Bob and Sum Chin Sean. There is no Internet draft for that at present.
- o Text for Section 4 was provided by Ana Minabiru, Carles Gomez, Laurent Toutain, Josep Paradells and Jon Crowcroft in [I-D.minaburo-lpwan-gap-analysis]. Additional text from that draft is also used elsewhere above.

The full list of contributors are:

Jon Crowcroft
University of Cambridge
JJ Thomson Avenue
Cambridge, CB3 0FD
United Kingdom

Email: jon.crowcroft@cl.cam.ac.uk

Carles Gomez
UPC/i2CAT
C/Esteve Terradas, 7

Castelldefels 08860
Spain

Email: carlesgo@entel.upc.edu

Bob Heile
Wi-Sun Alliance
11 Robert Toner Blvd, Suite 5-301
North Attleboro, MA 02763
USA

Phone: +1-781-929-4832
Email: bheile@ieee.org

Ana Minaburo
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

Josep PARadells
UPC/i2CAT
C/Jordi Girona, 1-3
Barcelona 08034
Spain

Email: josep.paradells@entel.upc.edu

Benoit Ponsard
SIGFOX
425 rue Jean Rostand
Labege 31670
France

Email: Benoit.Ponsard@sigfox.com
URI: <http://www.sigfox.com/>

Antti Ratilainen
Ericsson
Hirsalantie 11
Jorvas 02420

Finland

Email: antti.ratilainen@ericsson.com

Chin-Sean SUM
Wi-Sun Alliance
20, Science Park Rd
Singapore 117674

Phone: +65 6771 1011
Email: sum@wi-sun.org

Laurent Toutain
Institut MINES TELECOM ; TELECOM Bretagne
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@telecom-bretagne.eu

Alper Yegin
Actility
Paris, Paris
FR

Email: alper.yegin@actility.com

Juan Carlos Zuniga
SIGFOX
425 rue Jean Rostand
Labege 31670
France

Email: JuanCarlos.Zuniga@sigfox.com
URI: <http://www.sigfox.com/>

8. Acknowledgments

Thanks to all those listed in Section 7 for the excellent text.
Errors in the handling of that are solely the editor's fault.

In addition to the contributors above, thanks are due to Arun (arun@acklio.com), Dan Garcia Carrillo, Paul Duffy, Russ Housley, Thad Guidry, Jiazi Yi, for comments.

Alexander Pelov and Pascal Thubert were the LPWAN WG chairs while this document was developed.

Stephen Farrell's work on this memo was supported by Pervasive Nation, the Science Foundation Ireland's CONNECT centre national IoT network. <<https://connectcentre.ie/pervasive-nation/>>

9. Informative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, DOI 10.17487/RFC2904, August 2000, <<https://www.rfc-editor.org/info/rfc2904>>.
- [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, DOI 10.17487/RFC3095, July 2001, <<https://www.rfc-editor.org/info/rfc3095>>.
- [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, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <<https://www.rfc-editor.org/info/rfc3963>>.

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [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, <<https://www.rfc-editor.org/info/rfc4861>>.
- [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, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <<https://www.rfc-editor.org/info/rfc5216>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [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, <<https://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, <<https://www.rfc-editor.org/info/rfc6775>>.

- [RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, DOI 10.17487/RFC6961, June 2013, <<https://www.rfc-editor.org/info/rfc6961>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [RFC7668] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", RFC 7668, DOI 10.17487/RFC7668, October 2015, <<https://www.rfc-editor.org/info/rfc7668>>.
- [I-D.farrell-lpwan-lora-overview]
Farrell, S. and A. Yegin, "LoRaWAN Overview", draft-farrell-lpwan-lora-overview-01 (work in progress), October 2016.
- [I-D.minaburo-lpwan-gap-analysis]
Minaburo, A., Gomez, C., Toutain, L., Paradells, J., and J. Crowcroft, "LPWAN Survey and GAP Analysis", draft-minaburo-lpwan-gap-analysis-02 (work in progress), October 2016.
- [I-D.zuniga-lpwan-sigfox-system-description]
Zuniga, J. and B. PONSARD, "SIGFOX System Description", draft-zuniga-lpwan-sigfox-system-description-03 (work in progress), June 2017.
- [I-D.ratilainen-lpwan-nb-iot]
Ratilainen, A., "NB-IoT characteristics", draft-ratilainen-lpwan-nb-iot-00 (work in progress), July 2016.
- [I-D.garcia-dime-diameter-lorawan]
Garcia, D., Lopez, R., Kandasamy, A., and A. Pelov, "LoRaWAN Authentication in Diameter", draft-garcia-dime-diameter-lorawan-00 (work in progress), May 2016.
- [I-D.garcia-radext-radius-lorawan]
Garcia, D., Lopez, R., Kandasamy, A., and A. Pelov, "LoRaWAN Authentication in RADIUS", draft-garcia-radext-radius-lorawan-03 (work in progress), May 2017.

- [TGPP36300] 3GPP, "TS 36.300 v13.4.0 Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", 2016,
 <http://www.3gpp.org/ftp/Specs/2016-09/Rel-14/36_series/>.

- [TGPP36321] 3GPP, "TS 36.321 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification", 2016.

- [TGPP36322] 3GPP, "TS 36.322 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification", 2016.

- [TGPP36323] 3GPP, "TS 36.323 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification (Not yet available)", 2016.

- [TGPP36331] 3GPP, "TS 36.331 v13.2.0 Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification", 2016.

- [TGPP36201] 3GPP, "TS 36.201 v13.2.0 - Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description", 2016.

- [TGPP23720] 3GPP, "TR 23.720 v13.0.0 - Study on architecture enhancements for Cellular Internet of Things", 2016.

- [TGPP33203] 3GPP, "TS 33.203 v13.1.0 - 3G security; Access security for IP-based services", 2016.

- [fcc_ref] "FCC CFR 47 Part 15.247 Telecommunication Radio Frequency Devices - Operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz.", June 2016.

[etsi_ref]
"ETSI EN 300-220 (Parts 1 and 2): Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW", May 2016.

[arib_ref]
"ARIB STD-T108 (Version 1.0): 920MHz-Band Telemeter, Telecontrol and data transmission radio equipment.", February 2012.

[LoRaSpec]
LoRa Alliance, "LoRaWAN Specification Version V1.0.2", July 2016, <http://portal.lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=1398>.

[LoRaSpec1.0]
LoRa Alliance, "LoRaWAN Specification Version V1.0", Jan 2015, <<https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>>.

[ANSI-4957-000]
ANSI, TIA-4957.000, "Architecture Overview for the Smart Utility Network", May 2013, <https://global.ihs.com/doc_detail.cfm?%26rid=TIA%26item_s_key=00606368>.

[ANSI-4957-210]
ANSI, TIA-4957.210, "Multi-Hop Delivery Specification of a Data Link Sub-Layer", May 2013, <https://global.ihs.com/doc_detail.cfm?%26csf=TIA%26item_s_key=00601800>.

[wisun-pressie1]
Phil Beecher, Chair, Wi-SUN Alliance, "Wi-SUN Alliance Overview", March 2017, <<http://indiasmartgrid.org/event2017/10-03-2017/4.%20Roundtable%20on%20Communication%20and%20Cyber%20Security/1.%20Phil%20Beecher.pdf>>.

[wisun-pressie2]
Bob Heile, Director of Standards, Wi-SUN Alliance, "IETF97 Wi-SUN Alliance Field Area Network (FAN) Overview", November 2016, <<https://www.ietf.org/proceedings/97/slides/slides-97-lpwan-35-wi-sun-presentation-00.pdf>>.

[IEEE-802-15-4]

"IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Standard 802.15.4, 2015, <<https://standards.ieee.org/findstds/standard/802.15.4-2015.html>>.

[IEEE-802-15-9]

"IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams", IEEE Standard 802.15.9, 2016, <<https://standards.ieee.org/findstds/standard/802.15.9-2016.html>>.

[etsi_unb]

"ETSI TR 103 435 System Reference document (SRdoc); Short Range Devices (SRD); Technical characteristics for Ultra Narrow Band (UNB) SRDs operating in the UHF spectrum below 1 GHz", February 2017.

Appendix A. Changes

A.1. From -00 to -01

- o WG have stated they want this to be an RFC.
- o WG clearly want to keep the RF details.
- o Various changes made to remove/resolve a number of editorial notes from -00 (in some cases as per suggestions from Ana Minaburo)
- o Merged PR's: #1...
- o Rejected PR's: #2 (change was made to .txt not .xml but was replicated manually by editor)
- o Github repo is at: <https://github.com/sftcd/lpwan-ov>

A.2. From -01 to -02

- o WG seem to agree with editor suggestions in slides 13-24 of the presentation on this topic given at IETF98 (See: <https://www.ietf.org/proceedings/98/slides/slides-98-lpwan-aggregated-slides-07.pdf>)
- o Got new text wrt Wi-SUN via email from Paul Duffy and merged that in
- o Reflected list discussion wrt terminology and "end-device"

- o Merged PR's: #3...
- A.3. From -02 to -03
- o Editorial changes and typo fixes thanks to Fred Baker running something called Grammerly and sending me it's report.
 - o Merged PR's: #4, #6, #7...
 - o Editor did an editing pass on the lot.
- A.4. From -03 to -04
- o Picked up a PR that had been wrongly applied that expands UE
 - o Editorial changes wrt LoRa suggested by Alper
 - o Editorial changes wrt SIGFOX provided by Juan-Carlos
- A.5. From -04 to -05
- o Handled Russ Housley's WGLC review.
 - o Handled Alper Yegin's WGLC review.
- A.6. From -05 to -06
- o More Alper comments:-)
 - o Added some more detail about sigfox security.
 - o Added Wi-SUN changes from Charlie Perkins

Author's Address

Stephen Farrell (editor)
Trinity College Dublin
Dublin 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie