

6TiSCH
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

D. Dujovne, Ed.
Universidad Diego Portales
LA. Grieco
Politecnico di Bari
MR. Palattella
University of Luxembourg
N. Accettura
University of California Berkeley
March 21, 2016

6TiSCH 6top Scheduling Function Zero (SF0)
draft-dujovne-6tisch-6top-sf0-01

Abstract

This document defines a 6top Scheduling Function called "Scheduling Function Zero" (SF0). SF0 dynamically adapts the number of reserved cells between neighbor nodes, based on the currently allocated bandwidth and the neighbour nodes' requirements. Neighbor nodes negotiate in a distributed neighbor-to-neighbor basis the cell(s) to be added/deleted. SF0 uses the 6P signaling messages to add/delete cells in the schedule. Some basic rules for deciding when to add/delete cells and for selecting the cells to be added/deleted within the schedule are also provided.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 22, 2016.

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. Scheduling Function Identifier	3
3. Rules for Adding/Deleting Cells	3
3.1. SF0 Triggering Events	3
3.2. SF0 Bandwidth Estimation Algorithm	3
3.3. SF0 Allocation Policy	4
4. Rules for CellList	5
5. 6P Timeout Value	6
6. Meaning of Metadata Information	6
7. Node Behavior at Boot	6
8. Relocating Cells	6
9. Forced Cell Deletion Policy	7
10. 6P Error Handling	7
11. Examples	7
12. Implementation Status	7
13. Security Considerations	8
14. IANA Considerations	8
15. Acknowledgments	8
16. References	8
16.1. Normative References	8
16.2. Informative References	9
Authors' Addresses	9

1. Introduction

This document defines the Scheduling Function for the 6top sublayer [I-D.wang-6tisch-6top-sublayer] called "Scheduling Function Zero" (SF0).

This document addresses the requirements for a scheduling function listed in [I-D.wang-6tisch-6top-sublayer], Section 4.2, and follows the recommended outline from Section 4.3.

2. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of SF0 is IANA_SFID_SF0.

3. Rules for Adding/Deleting Cells

A node running SF0 determines when to add/delete cells in a three-step process:

1. It waits for a triggering event (Section 3.1).
2. It applies the Bandwidth Estimation Algorithm (BEA) for a particular neighbor to determine how much bandwidth is required to that neighbor (Section 3.2).
3. It applies the Allocation Policy to compare the number of required cells to the number of already scheduled cells, and determine the number of cells to add/delete (Section 3.3).

3.1. SF0 Triggering Events

We RECOMMEND SF0 to be triggered at least by the following events:

1. If the Remaining Available Bandwidth (RAB) is less than the Minimum Remaining Bandwidth (MRB)
2. If there is any New Incoming Bandwidth Requirements from neighbour nodes (NIBR)

This allows SF0 to be triggered by any change in local node bandwidth and/or incoming bandwidth. The exact mechanism of when SF0 is triggered is implementation-specific.

3.2. SF0 Bandwidth Estimation Algorithm

The Bandwidth Estimation Algorithm takes into account the sum of the incoming bandwidth requirements from the neighbour nodes and the used outgoing bandwidth. This allows the node to estimate the total outgoing bandwidth requirement. As a consequence, the Bandwidth Estimation Algorithm for SF0 follows the steps described below:

1. Collect the New Incoming Bandwidth Requirements from neighbour nodes (NIBR)
2. Obtain the Current Outgoing Bandwidth Usage (COBU)
3. Obtain the number of Current Scheduled Bandwidth (CSB)
4. Calculate the New Outgoing Bandwidth (NOB) as: $NOB = COBU + NIBR$
5. Calculate the Remaining Available Bandwidth (RAB) as $RAB = CSB - NOB$

6. If the RAB is less than the Minimum Remaining Bandwidth (MRB),
Add MRB to the NOB: $NOB = NOB + MRB$
7. Submit the request to the allocation policy
8. Return to step 1 and wait for a triggering event.

3.3. SF0 Allocation Policy

The "Allocation Policy" is the set of rules used by SF0 to decide when to add/delete cells to a particular neighbor to satisfy the bandwidth requirements.

SF0 uses the following parameters:

SCHEDULEDCELLS: The number of cells scheduled from the current node to a particular neighbor.

REQUIREDCELLS: The number of cells calculated by the Bandwidth Estimation Algorithm from the current node to that neighbor.

SF0THRESH: Threshold parameter introducing cell over-provisioning in the allocation policy. It is a non-negative value expressed as number of cells. The definition of this value is implementation-specific; however, it is RECOMMENDED a SF0THRESH value of 3 cells. A setting of $SF0THRESH > 0$ will cause the node to allocate at least SF0THRESH cells to each of its' neighbours.

The SF0 allocation policy compares REQUIREDCELLS with SCHEDULEDCELLS and decides to add/delete cells taking into account SF0THRESH. This is illustrated in Figure 1.

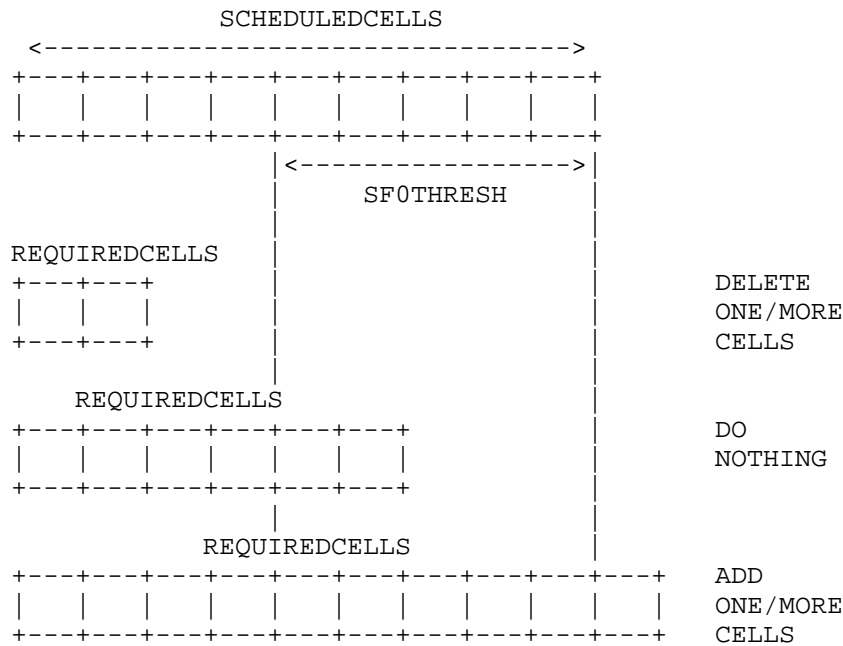


Figure 1: The SF0 Allocation Policy

1. If $\text{REQUIREDCELLS} < (\text{SCHEDULEDCELLS} - \text{SF0THRESH})$, delete one or more cells.
2. If $(\text{SCHEDULEDCELLS} - \text{SF0THRESH}) \leq \text{REQUIREDCELLS} \leq \text{SCHEDULEDCELLS}$, do nothing.
3. If $\text{SCHEDULEDCELLS} \leq \text{REQUIREDCELLS}$, add one or more cells.

When SF0THRESH equals 0, any discrepancy between REQUIREDCELLS and SCHEDULEDCELLS triggers an action to add/delete cells. Positive values of SF0THRESH reduce the number of 6P Transactions.

The Allocation Policy also translates the bandwidth requirement into cells according to their PDR. For example, if a cell with a 100% PDR is equivalent to 1Kbps, and the required bandwidth is 8Kbps, then, the number of scheduled cells will be 8. However, if two of the allocated cells have a 70% PDR, there number of scheduled cells will be 9.

4. Rules for CellList

When issuing a 6top ADD Request, SF0 executes the following sequence:

Whitelist case:

The Transaction Source node: Prepares the CellList field by selecting randomly the required cells, verifying that the slot offset and channel offset are not occupied.

The Transaction Destination node: Goes through the cells in the CellList in order, verifying whether there are no slotOffset conflicts.

Blacklist case:

The Transaction Source node: Prepares the CellList field by building a list of currently scheduled cells into the CellList.
The Transaction Destination node: Selects randomly the required cells, verifying that the slot offset and channel offset are not occupied from the ones on the CellList.

5. 6P Timeout Value

The 6P Timeout Value provided by SF0 allows the maximum number of TSCH link-layer retries. Given the TSCH parameters for the backoff mechanism, `macMinBE` and `macMaxBE`, and the length in seconds of the minimal Slotframe, `SM`, the timeout value is computed as: $\text{timeout} = (2^{(\text{macMaxBE}+1)} - 2^{\text{macMinBE}}) * \text{SM}$ TODO: Change general timeout to a timeout adapted to the schedule: SF to use the number of slots until the next scheduled cell.

6. Meaning of Metadata Information

The Metadata 16-bit field is used as follows:

BITS 0-7 [SLOTFRAME] are used to identify the slotframe number
BITS 8-14 are RESERVED
BIT 15 [WBLIST] is used to indicate that the CellList provided is a Whitelist (value=0) or a Blacklist (value=1).

TODO: length of the SlotFrame SHOULD be an integer multiple of the length of the minimal SlotFrame.

7. Node Behavior at Boot

In order to define a known state after the node is restarted, a CLEAR command is issued to each of the neighbour nodes to enable a new allocation process. TODO: Temporary cells from a pool for the join process.

8. Relocating Cells

SF0 uses Packet Delivery Rate (PDR) statistics to monitor the currently allocated cells for cell re-allocation (by changing their

slotOffset and/or channelOffset) when it finds out that the PDR of one or more softcells below 20% of the average PDR.

9. Forced Cell Deletion Policy

TODO: When all the cells are scheduled, we need a policy to free cells, for example, under alarm conditions or if a node disappears from the neighbour list.

10. 6P Error Handling

A node implementing SF0 handles a 6P Response depending on the Return Code it contains:

RC_SUCCESS:

If the number of elements in the CellList is the number of cells specified in the NumCells field of the 6P ALL Request, the operation is complete. The node does not take further action. If the number of elements in the CellList is smaller (possibly 0) than the number of cells specified in the NumCells field of the 6P ALL Request, the neighbor has received the request, but less than NumCells of the cells in the CellList were. In that case, the node MAY retry immediately with a different CellList if the amount of storage space permits, or build a new (random) CellList.

RC_ERR_VER: The node MUST NOT retry immediately. The node MAY add the neighbor node on a blacklist. The node MAY retry to contact this neighbor later.

RC_ERR_6OFID: The node MUST NOT retry immediately. The node MAY add the neighbor node on a blacklist. The node MAY retry to contact this neighbor later.

RC_ERR_NORESOURCES: Wait for a timeout and restart the scheduling process.

RC_ERR_BUSY: Issue a RESET command.

11. Examples

TODO

12. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was

supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

OpenWSN: This specification is implemented in the OpenWSN project [OpenWSN]. The authors of this document are collaborating with the OpenWSN community to gather feedback about the status and performance of the protocols described in this document. Results from that discussion will appear in this section in future revision of this specification.

13. Security Considerations

TODO

14. IANA Considerations

- o IANA_SFID_SF0

15. Acknowledgments

Thanks to Kris Pister for his contribution in designing the default Bandwidth Estimation Algorithm. Thanks to Qin Wang and Thomas Watteyne for their support in defining the interaction between SF0 and the 6top sublayer.

This work is partially supported by the Fondecyt 1121475 Project, the Inria-Chile "Network Design" group, and the IoT6 European Project (STREP) of the 7th Framework Program (Grant 288445).

16. References

16.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

[IEEE802154e]

IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

[IEEE802154]

IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks", June 2011.

16.2. Informative References

[RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<http://www.rfc-editor.org/info/rfc7554>>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.

[I-D.ietf-6tisch-terminology]

Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-07 (work in progress), March 2016.

[I-D.wang-6tisch-6top-sublayer]

Wang, Q. and X. Vilajosana, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-sublayer-04 (work in progress), November 2015.

[OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies, August 2012.

Authors' Addresses

Diego Dujovne (editor)
Universidad Diego Portales
Escuela de Informatica y Telecomunicaciones
Av. Ejercito 441
Santiago, Region Metropolitana
Chile

Phone: +56 (2) 676-8121
Email: diego.dujovne@mail.udp.cl

Luigi Alfredo Grieco
Politecnico di Bari
Department of Electrical and Information Engineering
Via Orabona 4
Bari 70125
Italy

Phone: 00390805963911
Email: a.grieco@poliba.it

Maria Rita Palattella
University of Luxembourg
Interdisciplinary Centre for Security, Reliability and Trust
4, rue Alphonse Weicker
Luxembourg L-2721
LUXEMBOURG

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

Nicola Accettura
University of California Berkeley
Berkeley Sensor & Actuator Center
490 Cory Hall
Berkeley, California 94720
USA

Email: nicola.accettura@eecs.berkeley.edu

6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: September 2, 2018

S. Duquennoy, Ed.
RISE SICS
X. Vilajosana
Universitat Oberta de Catalunya
T. Watteyne
Inria
March 1, 2018

6TiSCH Autonomous Scheduling Function (ASF)
draft-duquennoy-6tisch-asf-01

Abstract

This document defines a Scheduling Function called "ASF": the 6TiSCH Autonomous Scheduling Function. With ASF, nodes maintain their TSCH schedule based on local neighborhood knowledge, without any signaling after association. Hashes of the nodes' MAC address are used to deterministically derive the [slotOffset,channelOffset] location of cells in the TSCH schedule. Different traffic types (e.g. TSCH EB, RPL DIO, UDP etc.) are assigned to distinct slotframes, for isolation and flexible dimensioning. This approach provides over-provisioned schedules with low maintenance, in pursuit for simplicity rather than optimality.

Requirements Language

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

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 September 2, 2018.

Copyright Notice

Copyright (c) 2018 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. TEMPORARY EDITORIAL NOTES	3
2. Introduction	3
2.1. Application Domains	3
3. General Operation	4
3.1. Cell Coordinates	4
3.2. Types of Slotframes	4
3.2.1. Rendez-vous slotframe	4
3.2.2. Receiver-based slotframe	4
3.2.3. Sender-based slotframe	5
3.3. Conditional Cells	5
3.4. Interaction between Slotframes	5
4. Configuration	6
5. Scheduling Function Identifier	9
6. Rules for Adding/Deleting Cells	9
7. Rules for CellList	9
8. 6P Timeout Value	10
9. Rule for Ordering Cells	10
10. Meaning of the Metadata Field	10
11. Node Behavior at Boot	10
12. 6P Error Handling	10
13. Examples	11
14. [TEMPORARY] Implementation Status	11
15. Security Considerations	11
16. IANA Considerations	12
16.1. 6P Scheduling Function Identifiers 'ASF'	12
17. References	12
17.1. Normative References	12
17.2. Informative References	12
Appendix A. Contributors	13
Appendix B. Acknowledgments	13
Appendix C. [TEMPORARY] Changelog	13

Authors' Addresses	14
------------------------------	----

1. TEMPORARY EDITORIAL NOTES

This document is an Internet Draft, so work-in-progress by nature. It contains the following work-in-progress elements:

- o "TODO" statements are elements which have not yet been written by the authors for some reason (lack of time, ongoing discussions with no clear consensus, etc). The statement does indicate that the text will be written at some point.
- o "TEMPORARY" appendices are there to capture current ongoing discussions, or the changelog of the document. These appendices will be removed in the final text.
- o "IANA_*" identifiers are placeholders for numbers assigned by IANA. These placeholders are to be replaced by the actual values they represent after their assignment by IANA.
- o "RFCXXXX" refers to the RFC number of this specification, once published.
- o The string "REMARK" is put before a remark (questions, suggestion, etc) from an author, editor or contributor. These are on-going discussions at the time of writing, and will not be part of the final text.
- o This section will be removed in the final text.

2. Introduction

This document defines an autonomous Scheduling Function for the 6top sublayer [I-D.ietf-6tisch-6top-protocol], called "ASF". It is designed to operate without any runtime signaling, keeping the TSCH schedule consistent between neighbors at all times (slots for transmission and reception always match). ASF uses 6P solely for configuration at association time (6P SIGNAL) and for schedule inspection (6P STATUS and LIST). ASF isolates different traffic types into distinct slotframes, so as to avoid any disruption between MAC synchronization, control and application traffic.

ASF addresses all requirements listed in Section "Requirements for an SF" from [I-D.ietf-6tisch-6top-protocol]. The organization of this document follows section "Recommended Structure of an SF Specification" in [I-D.ietf-6tisch-6top-protocol]. This document follows the terminology defined in [I-D.ietf-6tisch-terminology].

2.1. Application Domains

ASF is primarily targeted at applications with random traffic flows, such as interactive CoAP traffic. Its main strength is its signaling-free nature, which ensures the slots installed at

neighboring nodes are consistent at all times. Its main weakness is its contention-based nature and its need to over-provision the schedule, rendering it unable to meet stringent latency and energy requirements. An example application domains is building instrumentation. ASF was evaluated experimentally and shown to achieve over 99.99% end-to-end delivery in 6TiSCH/RPL testbeds [Orchestra-SenSys].

3. General Operation

ASF uses multiple slotframes, each assigned to one particular type of traffic, e.g. TSCH EBS, RPL or UDP traffic. Nodes maintain the cells within the slotframes autonomously, based on the hash of either the source's or destination's MAC address. Each slotframe is uniquely assigned a set of channel offsets.

3.1. Cell Coordinates

Cell coordinates in ASF are either fixed or derived from a MAC address (depending on the slotframe type, see Section 3.2). To derive coordinates from a MAC (EUI-64) address, nodes MUST use the hash function provided at configuration time, see Section 4. One example hash function is SAX [SAX-DASFAA]. Let S_len be the length of slotframe S , and $S_channels$ be the set of channels assigned to slotframe S . The slot coordinates derived from a given MAC address are computed as follows:

```
slotOffset(MAC) = hash(MAC) % S_len
channelOffset(MAC) = S_channels[(hash(MAC) / L) % len(S_channels)]
```

3.2. Types of Slotframes

There are three different types of slotframes, described next. Section 4 provides full details on cell options and other aspects.

3.2.1. Rendez-vous slotframe

Contains a single contention-based rendez-vous cell, at coordinates [slot offset: 0; channel offset: 0]. This slotframe is equivalent to the 6TiSCH minimal schedule [RFC8180].

3.2.2. Receiver-based slotframe

One Rx cell: Coordinates computed as the hash of the node's own MAC address.

Multiple Tx cells: One Tx cell per neighbor. Coordinates computed as the hash of the neighbor's MAC address.

3.2.3. Sender-based slotframe

One Tx cell: Coordinates computed as the hash of the node's own MAC address.

Multiple Rx cells: One Rx cell per neighbor. Coordinates computed as the hash of the neighbor's MAC address.

3.3. Conditional Cells

In order to handle traffic bursts, ASF utilizes conditional cells. When a node has several frames in its queue for a given neighbor, it can set the [IEEE802154-2015] 'frame pending' bit in unicast transmissions to that neighbor. Cells at upcoming time offsets will be used to carry more frames. Note that collisions may happen on these conditional cells, which MUST therefore have the 'Shared' bit set.

Sender: A sender with multiple unicast frames in its queue for a given neighbor MAY send frames with the 'frame pending' bit set. After sending a unicast frame with the 'frame pending' bit set, if a link-layer Acknowledgment (ACK) is received, the sender immediately schedules a temporary Tx cell. Compared to the initial cell, the temporary cell has the same Link Options plus the 'Shared' bit, the same channel offset, and a time offset incremented by 1 (modulo the slotframe length). The next frame will be sent on this temporary cell, and may set the 'frame pending' bit again to signal more traffic to come. The procedure repeats until the transmit queue is empty, or until no acknowledgment is received for a frame.

Receiver: Upon receiving a unicast frame with the 'frame pending' bit set, the node first sends a link-layer ACK. It then schedules a temporary Rx cell, with same Link Options, same channel offset, and time offset incremented by 1. If, in the new cell, it receives a unicast frame with the 'frame pending' bit set, it continues scheduling additional Rx cells to receive subsequent frames.

3.4. Interaction between Slotframes

ASF is expected to maintain multiple slotframes, each dedicated to a different traffic type. As the slotframes repeat over time, cells from different slotframes overlap periodically. In case a node has multiple cells scheduled at the same time, the precedence rules from [IEEE802154-2015] apply. In order to distribute cell overlap uniformly, it is RECOMMENDED to select slotframe lengths that are co-primes.

4. Configuration

An ASF configuration consists of a series of slotframes with attributes. ASF uses the 6P SIGNAL command (format in Figure 1) to disseminate its configuration. SIGNAL commands are directly included as IETF IE in each EB, so that nodes learn the ASF configuration directly at join-time.

6TiSCH EBs are not secured. For applications that require the ASF schedule to be sent securely, the ASF SIGNAL command MAY be sent instead in separate data broadcast packets, after join-time. To summarize, there are two cases:

Initial EB includes ASF SIGNAL: The node configures itself to run the ASF configuration provided.

Initial EB does not include ASF SIGNAL: The node runs the 6TiSCH minimal schedule ([RFC8180]) at association. When later receiving a packet with ASF SIGNAL, the node replaces the 6TiSCH minimal schedule with the ASF configuration provided.

Figure 1 describes the format of the ASF SIGNAL command.

```

                                1                2                3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| # Slotframes | Slotframe list ...
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 1: Format of the ASF SIGNAL command.

Where:

Slotframes: The number of ASF slotframes

Slotframe list: The list of slotframes, with format described in Figure 2

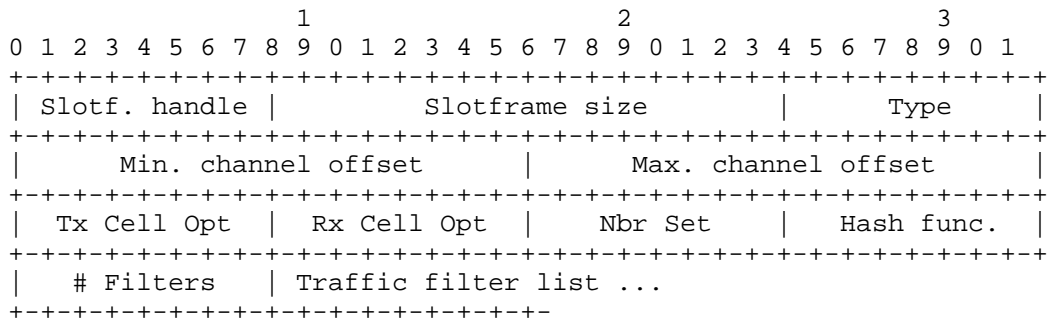


Figure 2: Format of the ASF SIGNAL slotframe descriptor.

Figure 2 shows the format of a slotframe descriptor, where:

Slotf. handle: The IEEE 802.15.4 slotframe handle (8 bits)
Slotframe size: The IEEE 802.15.4 slotframe size (16 bits)
Type: The ASF slotframe type. The set of possible values for this field is presented in Figure 3. The different slotframe types are described in Section 3.2.
Min. channel offset: ASF slotframes are assigned a channel offset range. This defines the lower bound for the range.
Max. channel offset: ASF slotframes are assigned a channel offset range. This defines the upper bound for the range.
Tx Cell Opt: The options to be used for the Tx Cells, if any.
Rx Cell Opt: The options to be used for the Rx Cells, if any.
Nbr Set: The set of neighbors for which Cells are instantiated. The set of possible values for this field is presented in Figure 4.
Hash func.: The hash function used to compute cell coordinates from a node's EUI-64 address, as defined in Section 3.1. The set of possible values for this field is presented in Figure 5.
Filters: ASF slotframes are assigned a subset of the traffic each. One or several traffic filters will be applied to only select packets with the intended properties. When there are several traffic filters, they are combined with a OR, i.e., packets that satisfy any of the filters will be sent on the slotframe. This field defines how many filters are in place. The filter descriptions follow inline, with format defined in Figure 6.

Num.	Description
0	Rendez-vous slotframe
1	Receiver-based slotframe
2	Sender-based slotframe
128--255	Reserved

Figure 3: Field: types of slotframes.

Num.	Description
0	Empty set
1	All TSCH time sources
2	All RPL parents
3	The RPL preferred parent
4	All IPv6 NDP neighbors
128--255	Reserved

Figure 4: Field: neighbor set.

Num.	Description
0	SAX (Shift-Add-XOR)
1--255	Reserved

Figure 5: Field: Hash function

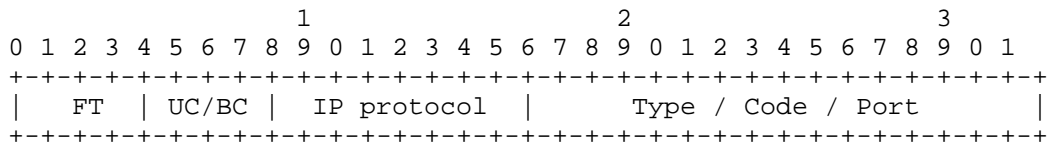


Figure 6: Format of the ASF SIGNAL traffic filters.

The fields for traffic filter descriptions (Figure 6) are described next:

FT: The IEEE 802.15.4 frame type. Examples; 0: Beacon; 1: Data.
UC/BC: Unicast/broadcast nature. Possible values; 0: unicast; 1: broadcast; 2: any.
IP protocol: The IP protocol number. Examples; 0x3a: ICMPv6; 0x11: UDP. A value of 0x00 ignores this field.
Type / Code / Port: In the case of ICMPv6: the Type (first 8 bits) followed by the Code (last 8 bits). In case of UDP or TCP: the 16-bit port. A value of 0x00 ignores this field.

Example filters are given next:

All TSCH EBs: FT: 0 (beacon), UC/BC: 1 (broadcast), IP protocol: 0, Type: 0.
All RPL Traffic: FT: 1 (data), UC/BC: 2 (unicast and broadcast), IP protocol: 0x3a (ICMPv6), Type: 0x9b (RPL), Code: 0x00
RPL Unicast DIO: FT: 1 (data), UC/BC: 0 (unicast), IP protocol: 0x3a (ICMPv6), Type: 0x9b (RPL), Code: 0x01 (DIO)
UDP port 5683: FT: 1 (data), UC/BC: 2 (unicast and broadcast), IP protocol: 0x11 (UDP), Port: 5683

5. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of ASF is IANA_SFID_ASF.

6. Rules for Adding/Deleting Cells

ASF nodes maintain their cells autonomously, and do not use 6P ADD nor DELETE.

7. Rules for CellList

For the 6P LIST command, ASF uses the default CellList field format defined in Section 4.2.4 [TODO: update if needed] of [I-D.ietf-6tisch-6top-protocol].

8. 6P Timeout Value

The timeout is of low criticality in ASF as 6P Requests are only used for schedule inspection, not for cell addition/removal. The RECOMMENDED timeout value in slots is:

$$2^{(\text{macMaxBe}+2)} * \text{SlotframeD_len}$$

which is an upper bound of the maximum time spent in transmission attempts of a 6P Request and Response, over slotframeD (where 6P traffic is sent). The upper bound is conservative, giving extra time for time spent in packet queues.

9. Rule for Ordering Cells

Cells are ordered by increasing slotframe handle, then by timeslot, then channel offset.

10. Meaning of the Metadata Field

The Metadata 16-bit field is used as follows: Figure 7 shows the format of the Metadata field, where:

- o Slotframe: is used to identify a slotframe by its handle.
- o Bits 8-15 are reserved.

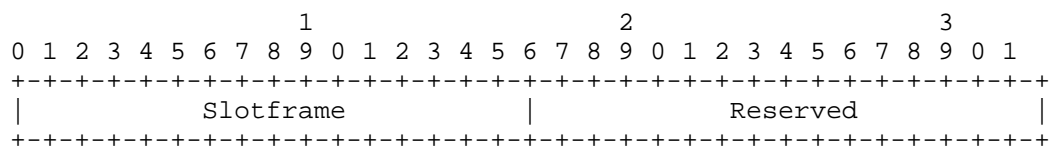


Figure 7: Format of the Metadata Field.

11. Node Behavior at Boot

At boot, nodes start with an empty schedule. When associating, they configure their schedule with the 6P ASF SIGNAL command, which is included either in the initial EB or later packets, as described in Section 4.

12. 6P Error Handling

ASF only uses 6P commands COUNT and LIST. In case of error on STATUS or LIST, the node MAY retry to contact this neighbor after the 6P timeout.

13. Examples

TODO

14. [TEMPORARY] Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Contiki: The mechanism behind this specification is implemented in the Contiki project [Contiki]. Adjustments to exactly match this specification are in progress. The mechanism was evaluated experimentally in large-scale testbeds in [Orchestra-SenSys].

15. Security Considerations

At run-time, ASF is not threatened by attacks on 6P messages as it operates without signaling. However, it bases its TSCH schedule on external information, namely: (1) the identify of the current TSCH time source and (2) the MAC address of its neighbors. ASF relies on link-layer security to ensure the integrity of the above information.

At configuration time, ASF relies on a 6P SIGNAL command. This command MAY be secured as described in Section 4. When this command is not secured, the security of the network is equivalent to that of the 6TiSCH minimal configuration ([RFC8180]). That is, the network schedule is propagated directly through EBs.

16. IANA Considerations

16.1. 6P Scheduling Function Identifiers 'ASF'

This document adds the following number to the "6P Scheduling Function Identifiers" registry defined by [I-D.ietf-6tisch-6top-protocol]:

SFID	Name	Reference
IANA_6TiSCH_SFID_ASF	Autonomous Scheduling Function (ASF)	RFCXXXX

Figure 8: 6P Scheduling Function Identifiers 'ASF'.

17. References

17.1. Normative References

- [IEEE802154-2015]
IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

17.2. Informative References

- [Contiki] Dunkels, A., Lignan, A., Thebaudeau, B., Quattlebaum, R., Rosendal, F., Oikonomou, G., Deru, L., Alvira, M., Tsiftes, N., Schmidt, O., and S. Duquennoy, "The Contiki Open Source OS for the Internet of Things", <https://github.com/contiki-os/contiki>, November 2016.
- [I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-09 (work in progress), October 2017.

[I-D.ietf-6tisch-terminology]

Palattella, M., Thubert, P., Watteyne, T., and Q. Wang,
"Terminology in IPv6 over the TSCH mode of IEEE
802.15.4e", draft-ietf-6tisch-terminology-09 (work in
progress), June 2017.

[Orchestra-SenSys]

Duquennoy, S., Al Nahas, B., Landsiedel, O., and T.
Watteyne, "Orchestra: Robust Mesh Networks Through
Autonomously Scheduled TSCH", ACM SenSys 2015 , November
2015.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running
Code: The Implementation Status Section", RFC 6982,
DOI 10.17487/RFC6982, July 2013,
<<https://www.rfc-editor.org/info/rfc6982>>.

[RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal
IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH)
Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180,
May 2017, <<https://www.rfc-editor.org/info/rfc8180>>.

[SAX-DASFAA]

Ramakrishna, M. and J. Zobel, "Performance in Practice of
String Hashing Functions", DASFAA , 1997.

Appendix A. Contributors

Beshr Al Nahas (Chalmers University, beshr@chalmers.se) and Olaf
Landsiedel (Chalmers University, olafl@chalmers.se) contributed to
the design and evaluation of ASF.

Appendix B. Acknowledgments

TODO people

TODO projects

Appendix C. [TEMPORARY] Changelog

o draft-duquennoy-6tisch-asf-01

- * Defines ASF configuration parameters and procedure;
- * Defines packet format to disseminate configurations (6P
signal);
- * Defines burst mode (conditional cells based on 'frame pending'
bit);
- * Makes Hash function configurable (SAX remains default).

o draft-duquennoy-6tisch-asf-00

* Initial draft.

Authors' Addresses

Simon Duquennoy (editor)
RISE SICS
Isafjordsgatan 22
164 29 Kista
Sweden

Email: simon.duquennoy@ri.se

Xavier Vilajosana
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Email: xvilajosana@uoc.edu

Thomas Watteyne
Inria
2 Rue Simone Iff
Paris
France

Email: thomas.watteyne@inria.fr

6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2018

Q. Wang, Ed.
Univ. of Sci. and Tech. Beijing
X. Vilajosana
Universitat Oberta de Catalunya
T. Watteyne
Analog Devices
June 20, 2018

6TiSCH Operation Sublayer Protocol (6P)
draft-ietf-6tisch-6top-protocol-12

Abstract

This document defines the IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) Operation Sublayer (6top) Protocol (6P), which enables distributed scheduling in 6TiSCH networks. 6P allows neighbor nodes to add/delete TSCH cells to one another. 6P is part of the 6TiSCH Operation Sublayer (6top), the next higher layer to the IEEE Std 802.15.4 TSCH medium access control layer. The 6top layer terminates the 6top Protocol defined in this document, and runs one or more 6top Scheduling Function(s). A 6top Scheduling Function (SF) decides when to add/delete cells, and triggers 6P Transactions. This document lists the requirements for an SF, but leaves the definition of SFs out of scope.

Requirements Language

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

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 December 22, 2018.

Copyright Notice

Copyright (c) 2018 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	3
2. 6TiSCH Operation Sublayer (6top)	4
2.1. Hard/Soft Cells	5
2.2. Using 6P with the Minimal 6TiSCH Configuration	5
3. 6top Protocol (6P)	6
3.1. 6P Transactions	6
3.1.1. 2-step 6P Transaction	7
3.1.2. 3-step 6P Transaction	9
3.2. Message Format	11
3.2.1. 6top Information Element (IE)	11
3.2.2. Generic 6P Message Format	11
3.2.3. 6P CellOptions	12
3.2.4. 6P CellList	15
3.3. 6P Commands and Operations	16
3.3.1. Adding Cells	16
3.3.2. Deleting Cells	18
3.3.3. Relocating Cells	19
3.3.4. Counting Cells	25
3.3.5. Listing Cells	26
3.3.6. Clearing the Schedule	28
3.3.7. Generic Signaling Between SFs	29
3.4. Protocol Functional Details	29
3.4.1. Version Checking	29
3.4.2. SFID Checking	30
3.4.3. Concurrent 6P Transactions	30
3.4.4. 6P Timeout	31
3.4.5. Aborting a 6P Transaction	31
3.4.6. SeqNum Management	31
3.4.7. Handling Error Responses	38

3.5. Security	38
4. Requirements for 6top Scheduling Functions (SF) Specification	38
4.1. SF Identifier (SFID)	38
4.2. Requirements for an SF specification	38
5. Security Considerations	39
6. IANA Considerations	39
6.1. IETF IE Subtype '6P'	40
6.2. 6TiSCH parameters sub-registries	40
6.2.1. 6P Version Numbers	40
6.2.2. 6P Message Types	41
6.2.3. 6P Command Identifiers	41
6.2.4. 6P Return Codes	42
6.2.5. 6P Scheduling Function Identifiers	43
6.2.6. 6P CellOptions bitmap	44
7. References	44
7.1. Normative References	45
7.2. Informative References	45
Appendix A. Recommended Structure of an SF Specification	46
Authors' Addresses	46

1. Introduction

All communication in a IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) network is orchestrated by a schedule [RFC7554]. The schedule is composed of cells, each identified by a [slotOffset,channelOffset]. This specification defines the 6TiSCH Operation Sublayer (6top) Protocol (6P), terminated by the 6TiSCH Operation sublayer (6top). 6P allows a node to communicate with a neighbor node to add/delete TSCH cells to one another. This results in distributed schedule management in a 6TiSCH network. The 6top layer terminates the 6top Protocol, and runs one or more 6top Scheduling Functions (SFs) that decide when to add/delete cells and trigger 6P Transactions. The SF is out of scope of this document but this document defines the requirements for an SF.

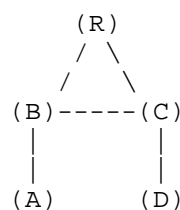


Figure 1: A simple 6TiSCH network.

The example network depicted in Figure 1 is used to describe the interaction between nodes. We consider the canonical case where node "A" issues 6P requests to node "B". We keep this example throughout

this document. Throughout the document, node A always represents the node that issues a 6P request; node B the node that receives this request.

We consider that node A monitors the communication cells it has in its schedule to node B:

- o If node A determines that the number of link-layer frames it is sending to node B per unit of time exceeds the capacity offered by the TSCH cells it has scheduled to node B, it triggers a 6P Transaction with node B to add one or more cells to the TSCH schedule of both nodes.
- o If the traffic is lower than the capacity, node A triggers a 6P Transaction with node B to delete one or more cells in the TSCH schedule of both nodes.
- o Node A MAY also monitor statistics to determine whether collisions are happening on a particular cell to node B. If this feature is enabled, node A communicates with node B to "relocate" the cell which undergoes collisions to a different [slotOffset,channelOffset] location in the TSCH schedule.

This results in distributed schedule management in a 6TiSCH network.

The 6top Scheduling Function (SF) defines when to add/delete a cell to a neighbor. Different applications require different SFs, so the SF is left out of scope of this document. Different SFs are expected to be defined in future companion specifications. A node MAY implement multiple SFs and run them at the same time. At least one SF MUST be running. The SFID field contained in all 6P messages allows a node to invoke the appropriate SF on a per-6P Transaction basis.

Section 2 describes the 6TiSCH Operation Sublayer (6top). Section 3 defines the 6top Protocol (6P). Section 4 provides guidelines on how to define an SF.

2. 6TiSCH Operation Sublayer (6top)

As depicted in Figure 2, the 6TiSCH Operation Sublayer (6top) is the next higher layer to the IEEE Std 802.15.4 TSCH medium access control (MAC) layer [IEEE802154]. We use "802.15.4" as a short version of "IEEE Std 802.15.4" in this document.

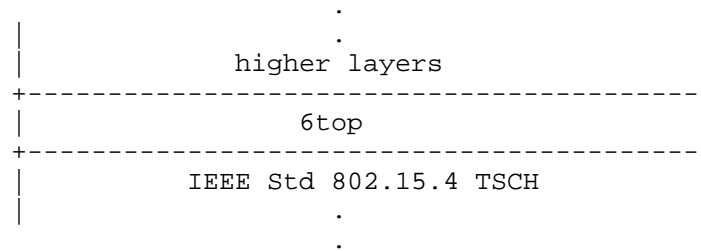


Figure 2: The 6top sublayer in the protocol stack.

The roles of the 6top sublayer are to:

- o Terminate the 6top Protocol (6P), which allows neighbor nodes to communicate to add/delete cells to one another.
- o Run one or multiple 6top Scheduling Functions (SFs), which define the rules that decide when to add/delete cells.

2.1. Hard/Soft Cells

Each cell in the schedule is either "hard" or "soft":

- o a soft cell can be read, added, deleted or updated by 6top.
- o a hard cell is read-only for 6top.

In the context of this specification, all the cells used by 6top are soft cells. Hard cells can be used for example when "hard-coding" a schedule [RFC8180].

2.2. Using 6P with the Minimal 6TiSCH Configuration

6P MAY be used alongside the Minimal 6TiSCH Configuration [RFC8180]. In this case, it is RECOMMENDED to use 2 slotframes, as depicted in Figure 3:

- o Slotframe 0 is used for traffic defined in the Minimal 6TiSCH Configuration. In Figure 3, Slotframe 0 is 5 slots long, but it can be shorter or longer.
- o 6P allocates cells from Slotframe 1. In Figure 3, Slotframe 1 is 10 slots long, but it can be shorter or longer.

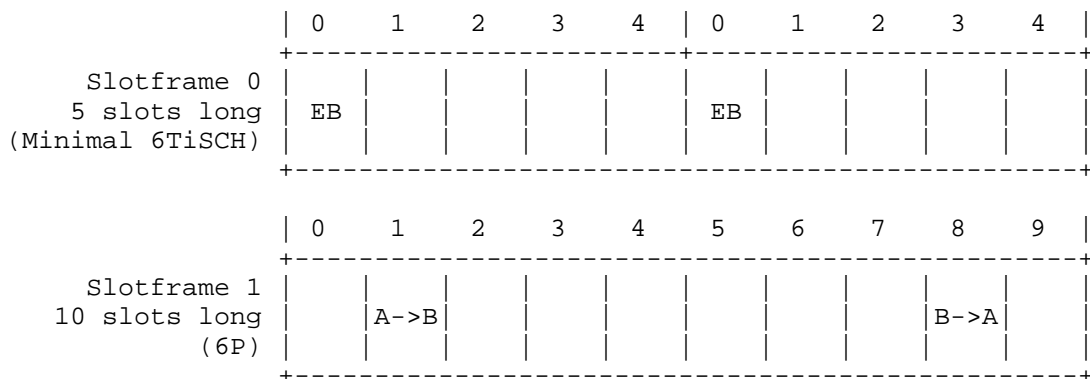


Figure 3: 2-slotframe structure when using 6P alongside the Minimal 6TiSCH Configuration.

The Minimal 6TiSCH Configuration cell SHOULD be allocated from a slotframe of higher priority than the slotframe used by 6P for dynamic cell allocation. This way, dynamically allocated cells cannot "mask" the cells used by the Minimal 6TiSCH Configuration. 6top MAY support additional slotframes; how to use additional slotframes is out of scope for this document.

3. 6top Protocol (6P)

The 6top Protocol (6P) enables two neighbor nodes to add/delete/relocate cells in their TSCH schedule. Conceptually, two neighbor nodes "negotiate" the location of the cells to add, delete, or relocate in their TSCH schedule.

3.1. 6P Transactions

We call "6P Transaction" a complete negotiation between two neighbor nodes. A particular 6P Transaction is executed between two nodes as a result of an action triggered by one SF. For a 6P Transaction to succeed, both nodes must use the same SF to handle the particular transaction. A 6P Transaction starts when a node wishes to add/delete/relocate one or more cells with one of its neighbors. A 6P Transaction ends when the cell(s) have been added/deleted/relocated in the schedule of both nodes, or when the 6P Transaction has failed.

6P messages exchanged between nodes A and B during a 6P Transaction SHOULD be exchanged on non-shared unicast cells ("dedicated" cells) between A and B. If no dedicated cells are scheduled between nodes A and B, shared cells MAY be used.

Keeping consistency between the schedules of the two neighbor nodes is important. A loss of consistency can cause loss of connectivity. One example is when node A has a transmit cell to node B, but node B does not have the corresponding reception cell. To verify consistency, neighbor nodes maintain a Sequence Number (SeqNum). Neighbor nodes exchange the SeqNum as part of each 6P Transaction to detect a possible inconsistency. This mechanism is explained in Section 3.4.6.2.

An implementation **MUST** include a mechanism to associate each scheduled cell with the SF that scheduled it. This mechanism is implementation-specific and out of scope of this document.

A 6P Transaction can consist of 2 or 3 steps. A 2-step transaction is used when node A selects the cells to be allocated. A 3-step transaction is used when node B selects the cells to be allocated. An SF **MUST** specify whether to use 2-step transactions, 3-step transactions, or both.

We illustrate 2-step and 3-step transactions using the topology in Figure 1.

3.1.1.1. 2-step 6P Transaction

Figure 4 shows an example 2-step 6P Transaction. In a 2-step transaction, node A selects the candidate cells. Several elements are left out to simplify understanding.

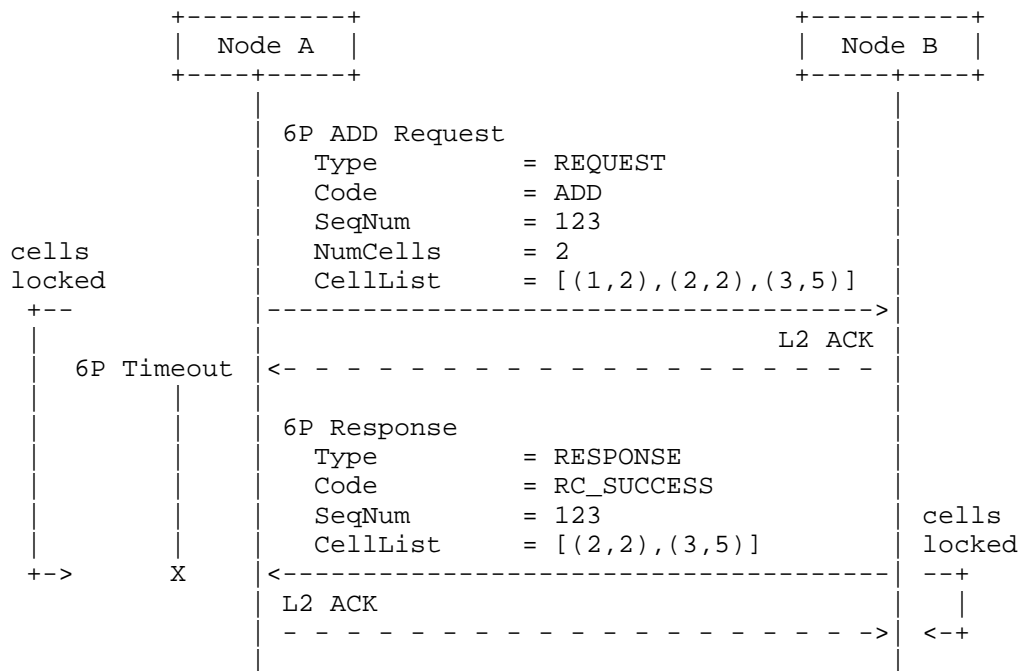


Figure 4: An example 2-step 6P Transaction.

In this example, the 2-step transaction occurs as follows:

1. The SF running on node A determines that 2 extra cells need to be scheduled to node B.
2. The SF running on node A selects candidate cells for node B to choose from. Node A MUST select at least as many candidate cells as the number of cells to add. Here, node A selects 3 candidate cells. Node A locks those candidate cells in its schedule until it receives a 6P response.
3. Node A sends a 6P ADD Request to node B, indicating it wishes to add 2 cells (the "NumCells" value), and specifying the list of 3 candidate cells (the "CellList" value). Each cell in the CellList is a [slotOffset,channelOffset] tuple. This 6P ADD Request is link-layer acknowledged by node B (labeled "L2 ACK" in Figure 4).
4. After having successfully sent the 6P ADD Request (i.e. receiving the link-layer acknowledgment), node A starts a 6P Timeout to abort the 6P Transaction in case no response is received from node B.
5. The SF running on node B selects 2 out of the 3 cells from the CellList of the 6P ADD Request. Node B locks those cells in its schedule until the transmission is successful (i.e. node B

receives a link-layer ACK from node A). Node B sends back a 6P Response to node A, indicating the cells it has selected. The response is link-layer acknowledged by node A.

6. Upon completion of this 6P Transaction, 2 cells from A to B have been added to the TSCH schedule of both nodes A and B.
7. An inconsistency in the schedule can happen if the 6P Timeout expires when the 6P Response is in the air, if the last link-layer ACK for the 6P Response is lost, or if one of the nodes is power cycled during the transaction. 6P provides an inconsistency detection mechanism described in Section 3.4.6.1 to cope with such situations.

3.1.2. 3-step 6P Transaction

Figure 5 shows an example 3-step 6P Transaction. In a 3-step transaction, node B selects the candidate cells. Several elements are left out to simplify understanding.

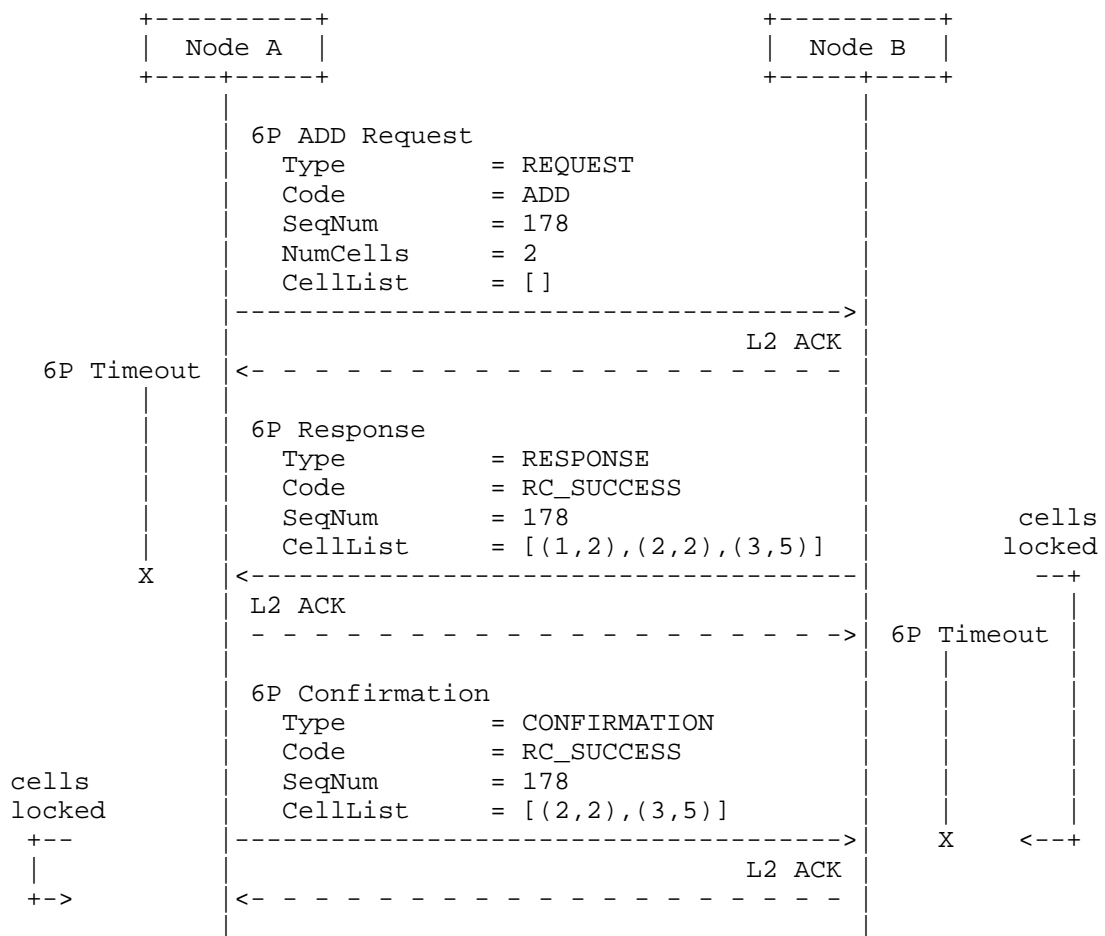


Figure 5: An example 3-step 6P Transaction.

In this example, the 3-step transaction occurs as follows:

1. The SF running on node A determines that 2 extra cells need to be scheduled to node B. The SF uses a 3-step transaction, so it does not select candidate cells.
2. Node A sends a 6P ADD Request to node B, indicating it wishes to add 2 cells (the "NumCells" value), with an empty "CellList". This 6P ADD Request is link-layer acknowledged by node B.
3. After having successfully sent the 6P ADD Request, node A starts a 6P Timeout to abort the transaction in case no 6P Response is received from node B.
4. The SF running on node B selects 3 candidate cells, and locks them. Node B sends back a 6P Response to node A, indicating the

- 3 cells it has selected. The response is link-layer acknowledged by node A.
5. After having successfully sent the 6P Response, node B starts a 6P Timeout to abort the transaction in case no 6P Confirmation is received from node A.
 6. The SF running on node A selects 2 cells from the CellList field in the 6P Response, and locks those. Node A sends back a 6P Confirmation to node B, indicating the cells it selected. The confirmation is link-layer acknowledged by node B.
 7. Upon completion of the 6P Transaction, 2 cells from A to B have been added to the TSCH schedule of both nodes A and B.
 8. An inconsistency in the schedule can happen if the 6P Timeout expires when the 6P Confirmation is in the air, if the last link-layer ACK for the 6P Confirmation is lost, or if one of the nodes is power cycled during the transaction. 6P provides an inconsistency detection mechanism described in Section 3.4.6.1 to cope with such situations.

3.2. Message Format

3.2.1. 6top Information Element (IE)

6P messages travel over a single hop. 6P messages are carried as payload of an 802.15.4 Payload Information Element (IE) [IEEE802154]. The messages are encapsulated within the Payload IE Header. The Group ID is set to the IETF IE value defined in [RFC8137]. The content is encapsulated by a SubType ID, as defined in [RFC8137].

Since 6P messages are carried in IEs, IEEE bit/byte ordering applies. Bits within each field in the 6top IE are numbered from 0 (leftmost and least significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are copied to the packet in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits (little endian).

This document defines the "6top IE", a SubType of the IETF IE defined in [RFC8137], with subtype ID IANA_6TOP_SUBIE_ID. The SubType Content of the "6top IE" is defined in Section 3.2.2. The length of the "6top IE" content is variable.

3.2.2. Generic 6P Message Format

All 6P messages follow the generic format shown in Figure 6.

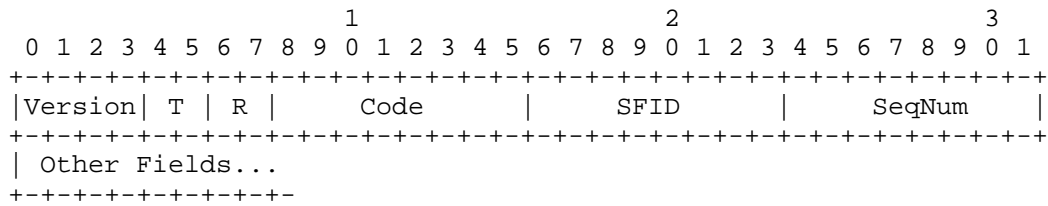


Figure 6: Generic 6P Message Format.

6P Version (Version): The version of the 6P protocol. Only version 0 is defined in this document. Future specifications may define further versions of the 6P protocol.

Type (T): Type of message. The message types are defined in Section 6.2.2.

Reserved (R): Reserved bits. These two bits SHOULD be set to zero when sending the message, and MUST be ignored upon reception.

Code: The Code field contains a 6P Command Identifier when the 6P message is of Type REQUEST. Section 6.2.3 lists the 6P command identifiers. The Code field contains a 6P return code when the 6P message is of Type RESPONSE or CONFIRMATION. Section 6.2.4 lists the 6P return codes. The same return codes are used in both 6P Response and 6P Confirmation messages.

6top Scheduling Function Identifier (SFID): The identifier of the SF to use to handle this message. The SFID is defined in Section 4.1.

SeqNum: Sequence number associated with the 6P Transaction, used to match the 6P Request, 6P Response and 6P Confirmation of the same 6P Transaction. The value of SeqNum MUST be different at each new 6P Request issued to the same neighbor and using the same SF. The SeqNum is also used to ensure consistency between the schedules of the two neighbors. Section 3.4.6 details how the SeqNum is managed.

Other Fields: The list of other fields and how they are used is detailed in Section 3.3.

6P Requests, 6P Response and 6P Confirmation messages for a same transaction MUST share the same Version, SFID and SeqNum values.

Future versions of the 6P Message SHOULD maintain the format of the 6P Version, Type and Code fields for backward compatibility.

3.2.3. 6P CellOptions

An 8-bit 6P CellOptions bitmap is present in the following 6P requests: ADD, DELETE, COUNT, LIST, RELOCATE. The format and meaning of this field MAY be redefined by the SF; the routine that parses this field is therefore associated with a specific SF.

- o In the 6P ADD request, the 6P CellOptions bitmap is used to specify what type of cell to add.
- o In the 6P DELETE request, the 6P CellOptions bitmap is used to specify what type of cell to delete.
- o In the 6P RELOCATE request, the 6P CellOptions bitmap is used to specify what type of cell to relocate.
- o In the 6P COUNT and the 6P LIST requests, the 6P CellOptions bitmap is used as a selector of a particular type of cells.

The content of the 6P CellOptions bitmap applies to all elements in the CellList field. The possible values of the 6P CellOptions are: TX = 1 (resp. 0) refers to macTxType = TRUE (resp. FALSE) in the macLinkTable of 802.15.4 [IEEE802154]. RX = 1 (resp. 0) refers to macRxType = TRUE (resp. FALSE) in the macLinkTable of 802.15.4. S = 1 (resp. 0) refers to macSharedType = TRUE (resp. FALSE) in the macLinkTable of 802.15.4. Section 6.2.6 contains the format of the 6P CellOptions bitmap, unless redefined by the SF. Figure 7 contains the meaning of the 6P CellOptions bitmap for the 6P ADD, DELETE, RELOCATE requests, unless redefined by the SF. Figure 8 contains the meaning of the 6P CellOptions bitmap for the 6P COUNT, LIST requests, unless redefined by the SF.

Note: assuming node A issues the 6P command to node B.

CellOptions Value	The type of cells B adds/deletes/relocates to its schedule when receiving a 6P ADD/DELETE/RELOCATE Request from A.
TX=0,RX=0,S=0	Invalid combination. RC_ERR is returned.
TX=1,RX=0,S=0	add/delete/relocate RX cells at B (TX cells at A)
TX=0,RX=1,S=0	add/delete/relocate TX cells at B (RX cells at A)
TX=1,RX=1,S=0	add/delete/relocate TX RX cells at B (and at A)
TX=0,RX=0,S=1	Invalid combination. RC_ERR is returned.
TX=1,RX=0,S=1	add/delete/relocate RX SHARED cells at B (TX SHARED cells at A)
TX=0,RX=1,S=1	add/delete/relocate TX SHARED cells at B (RX SHARED cells at A)
TX=1,RX=1,S=1	add/delete/relocate TX RX SHARED cells at B (and at A)

Figure 7: Meaning of the 6P CellOptions bitmap for the 6P ADD, DELETE, RELOCATE requests.

Note: assuming node A issues the 6P command to node B.

CellOptions Value	The type of cells B selects from its schedule when receiving a 6P COUNT or LIST Request from A, from all the cells B has scheduled with A
TX=0,RX=0,S=0	all cells
TX=1,RX=0,S=0	all cells marked as RX only
TX=0,RX=1,S=0	all cells marked as TX only
TX=1,RX=1,S=0	all cells marked as TX and RX only
TX=0,RX=0,S=1	all cells marked as SHARED (regardless of TX, RX)
TX=1,RX=0,S=1	all cells marked as RX and SHARED only
TX=0,RX=1,S=1	all cells marked as TX and SHARED only
TX=1,RX=1,S=1	all cells marked as TX and RX and SHARED

Figure 8: Meaning of the 6P CellOptions bitmap for the 6P COUNT, LIST requests.

The CellOptions is an opaque set of bits, sent unmodified to the SF. The SF MAY redefine the format and meaning of the CellOptions field.

3.2.4. 6P CellList

A CellList field MAY be present in a 6P ADD Request, a 6P DELETE Request, a 6P RELOCATE Request, a 6P Response, or a 6P Confirmation. It is composed of a concatenation of zero, one or more 6P Cells as defined in Figure 9. The content of the CellOptions field specifies the options associated with all cells in the CellList. This necessarily means that the same options are associated with all cells in the CellList.

A 6P Cell is a 4-byte field, its default format is:

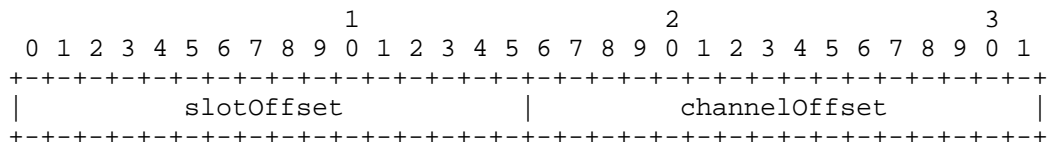


Figure 9: 6P Cell Format.

slotOffset: The slot offset of the cell.
channelOffset: The channel offset of the cell.

The CellList is an opaque set of bytes, sent unmodified to the SF. The length of the CellList field is implicit, and determined by the IE Length field of the Payload IE header as defined in 802.15.4. The SF MAY redefine the format of the CellList field; the routine that parses this field is therefore associated with a specific SF.

3.3. 6P Commands and Operations

3.3.1. Adding Cells

Cells are added by using the 6P ADD command. The Type field (T) is set to REQUEST. The Code field is set to ADD. Figure 10 defines the format of a 6P ADD Request.

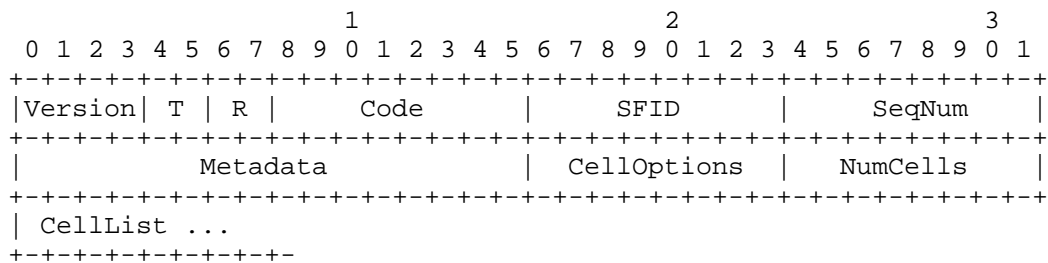


Figure 10: 6P ADD Request Format.

Metadata: Used as extra signaling to the SF. The contents of the Metadata field is an opaque set of bytes passed unmodified to the SF. The meaning of this field depends on the SF, and is out of scope of this document. For example, Metadata can specify in which slotframe to add the cells.

CellOptions: Indicates the options to associate with the cells to be added. If more than one cell is added (NumCells>1), the same options are associated with each one. This necessarily means that, if node A needs to add multiple cells with different options, it needs to initiate multiple 6P ADD Transactions.

NumCells: The number of additional cells node A wants to schedule to node B.

CellList: A list of 0 or multiple candidate cells. Its length is implicit and determined by the Length field of the Payload IE header.

Figure 11 defines the format of a 6P ADD Response and Confirmation.

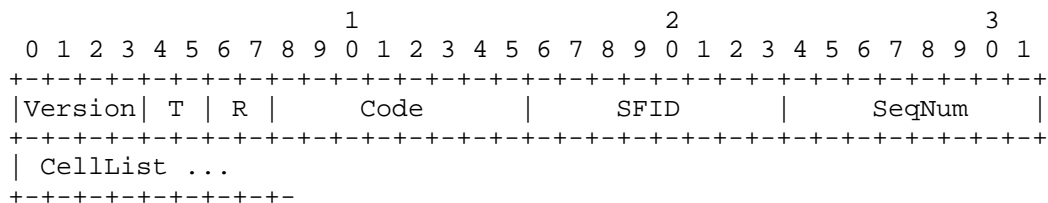


Figure 11: 6P ADD Response and Confirmation Formats.

CellList: A list of 0 or more 6P Cells.

Consider the topology in Figure 1 where the SF on node A decides to add NumCells cells to node B.

Node A's SF selects NumCandidate cells from its schedule. These are cells that are candidates to be scheduled with node B. The Celloptions field specifies the type of these cells. NumCandidate MUST be larger or equal to NumCells. How many cells node A selects (NumCandidate) and how that selection is done is specified in the SF and out of scope of this document. Node A sends a 6P ADD Request to node B which contains the Celloptions, the value of NumCells, and a selection of NumCandidate cells in the CellList. In case the NumCandidate cells do not fit in a single packet, this operation MUST be split into multiple independent 6P ADD Requests, each for a subset of the number of cells that eventually need to be added. In case of a 3-step transaction, the SF is responsible for ensuring that the returned candidate CellList fits into the 6P Response.

Upon receiving the request, node B checks whether the cellOptions are set to a valid value as noted by Figure 7. If this is not the case, a Response with code RC_ERR is returned. If the cells in the received CellList in node B is smaller than NumCells, Node B MUST return a 6P Response with RC_ERR_CELLLIST code. Otherwise, node B's SF verifies which of the cells in the CellList it can install in node B's schedule, following the specified CellOptions field. How that selection is done is specified in the SF and out of scope of this document. The verification can succeed (NumCells cells from the CellList can be used), fail (none of the cells from the CellList can be used), or partially succeed (fewer than NumCells cells from the CellList can be used). In all cases, node B MUST send a 6P Response with return code set to RC_SUCCESS, and which specifies the list of cells that were scheduled following the CellOptions field. That can contain NumCells elements (succeed), 0 elements (fail), or between 0 and NumCells elements (partially succeed).

Upon receiving the response, node A adds the cells specified in the CellList according to the CellOptions field.

3.3.2. Deleting Cells

Cells are deleted by using the 6P DELETE command. The Type field (T) is set to REQUEST. The Code field is set to DELETE. Figure 12 defines the format of a 6P DELETE Request.

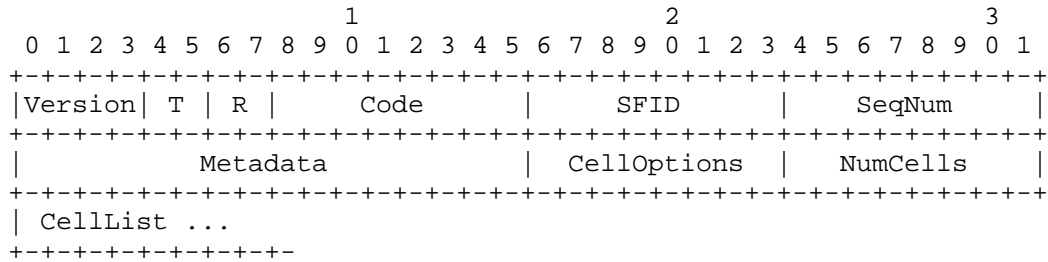


Figure 12: 6P DELETE Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1.

Its format is the same as that in the 6P ADD command, but its content could be different.

CellOptions: Indicates the options that need to be associated to the cells to delete. Only cells matching the CellOptions can be deleted.

NumCells: The number of cells from the specified CellList the sender wants to delete from the schedule of both sender and receiver.

CellList: A list of 0 or more 6P Cells. Its length is determined by the Length field of the Payload IE header.

Figure 13 defines the format of a 6P DELETE Response and Confirmation.

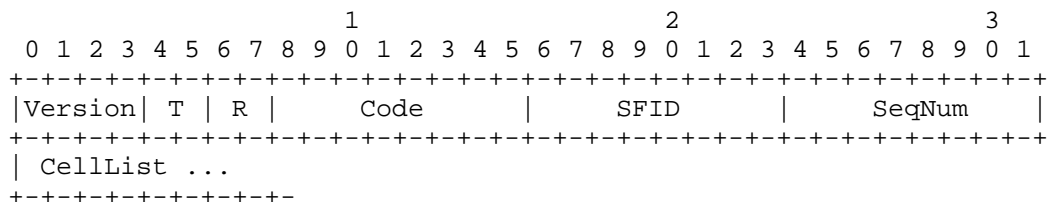


Figure 13: 6P DELETE Response and Confirmation Formats.

CellList: A list of 0 or more 6P Cells.

The behavior for deleting cells is equivalent to that of adding cells except that:

- o The nodes delete the cells they agree upon rather than adding them.
- o All cells in the CellList MUST already be scheduled between the two nodes and MUST match the CellOptions field. If node A puts cells in its CellList that are not already scheduled between the two nodes and match the CellOptions field, node B MUST reply with a RC_ERR_CELLLIST return code.
- o The CellList in a 6P Request (2-step transaction) or 6P Response (3-step transaction) MUST either be empty, contain exactly NumCells cells, or more than NumCells cells. The case where the CellList is not empty but contains fewer than NumCells cells is not supported. RC_ERR_CELLLIST code MUST be returned when the CellList contains fewer than NumCells cells. If the CellList is empty, the SF on the receiving node SHOULD choose NumCells cells with the sender from its schedule, which match the CellOption field, and delete them. If the CellList contains more than NumCells cells, the SF on the receiving node chooses exactly NumCells cells from the CellList to delete.

3.3.3. Relocating Cells

Cell relocation consists in moving a cell to a different [slotOffset,channelOffset] location in the schedule. The Type field (T) is set to REQUEST. The Code is set to RELOCATE. Figure 14 defines the format of a 6P RELOCATE Request.

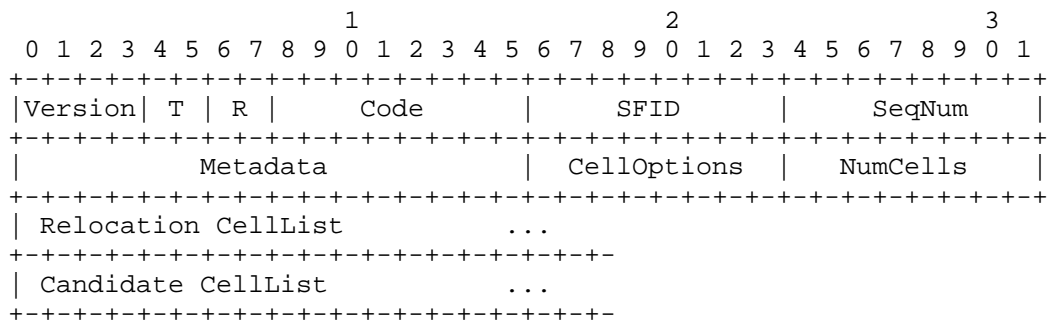


Figure 14: 6P RELOCATE Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1.

CellOptions: Indicates the options that need to be associated with cells to be relocated.

NumCells: The number of cells to relocate, which MUST be equal or greater than 1.

Relocation CellList: The list of NumCells 6P Cells to relocate.

Candidate CellList: A list of NumCandidate candidate cells for node B to pick from. NumCandidate MUST be 0, equal to NumCells, or

greater than NumCells. Its length is determined by the Length field of the Payload IE header.

In a 2-step 6P RELOCATE Transaction, node A specifies both the cells it needs to relocate, and the list of candidate cells to relocate to. The Relocation CellList MUST contain exactly NumCells entries. The Candidate CellList MUST contain at least NumCells entries (NumCandidate>=NumCells).

In a 3-step 6P RELOCATE Transaction, node A specifies only the cells it needs to relocate, but not the list of candidate cells to relocate to. The Candidate CellList MUST therefore be empty.

Figure 15 defines the format of a 6P RELOCATE Response and Confirmation.

1										2										3																			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Version										T R Code										SFID										SeqNum									
CellList ...																																							

Figure 15: 6P RELOCATE Response and Confirmation Formats.

CellList: A list of 0 or more 6P Cells.

Node A's SF wants to relocate NumCells cells. Node A creates a 6P RELOCATE Request, and indicates the cells it wants to relocate in the Relocation CellList. It also selects NumCandidate cells from its schedule as candidate cells to relocate the cells to, and puts those in the Candidate CellList. The Celloptions field specifies the type of the cell(s) to relocate. NumCandidate MUST be larger or equal to NumCells. How many cells it selects (NumCandidate) and how that selection is done is specified in the SF and out of scope of this document. Node A sends the 6P RELOCATE Request to node B.

Upon receiving the request, Node B checks if the length of the Candidate CellList is larger or equal to NumCells. Node B's SF verifies that all the cells in the Relocation CellList are scheduled with node A, and are associate the options specified in the Celloptions field. If either check fails, node B MUST send a 6P Response to node A with return code RC_ERR_CELLLIST. If both checks pass, node B's SF verifies which of the cells in the Candidate CellList it can install in its schedule. How that selection is done is specified in the SF and out of scope of this document. That verification on Candidate CellList can succeed (NumCells cells from

the Candidate CellList can be used), fail (none of the cells from the Candidate CellList can be used) or partially succeed (fewer than NumCells cells from the Candidate CellList can be used). In all cases, node B MUST send a 6P Response with return code set to RC_SUCCESS, and which specifies the list of cells that will be re-scheduled following the CellOptions field. That can contain NumCells elements (succeed), 0 elements (fail), between 0 and NumCells elements (partially succeed). If $N < \text{NumCells}$ cells appear in the CellList, this means the first N cells in the Relocation CellList have been relocated, the remainder have not.

Upon receiving the response with Code RC_SUCCESS, node A relocates the cells specified in Relocation CellList of its RELOCATE Request to the new locations specified in the CellList of the 6P Response, in the same order. In case the received return code is RC_ERR_CELLLIST, the transaction is aborted and no cell is relocated. In case of a 2-step transaction, Node B relocates the selected cells upon receiving the link-layer ACK for the 6P Response. In case of a 3-step transaction, Node B relocates the selected cells upon receiving the 6P Confirmation.

The SF SHOULD NOT relocate all cells between two nodes at the same time, which might result in the schedules of both nodes diverging significantly.

Figure 16 shows an example of a successful 2-step 6P RELOCATION Transaction.

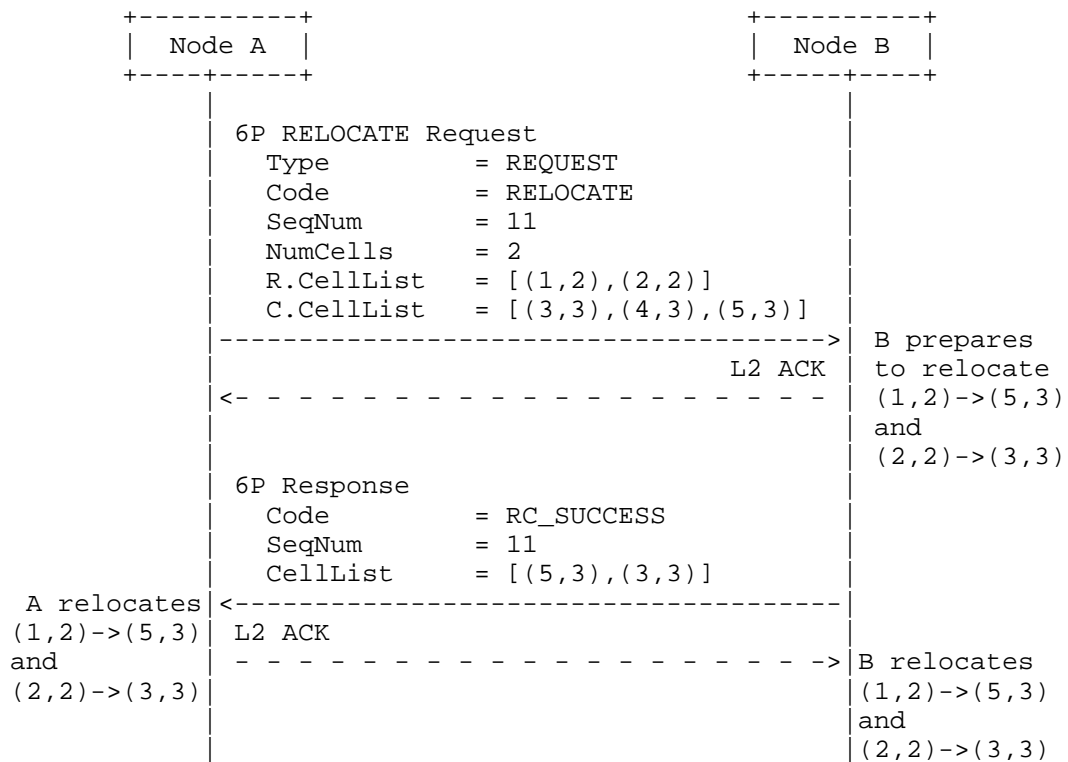


Figure 16: Example of a successful 2-step 6P RELOCATION Transaction.

Figure 17 shows an example of a partially successful 2-step 6P RELOCATION Transaction.

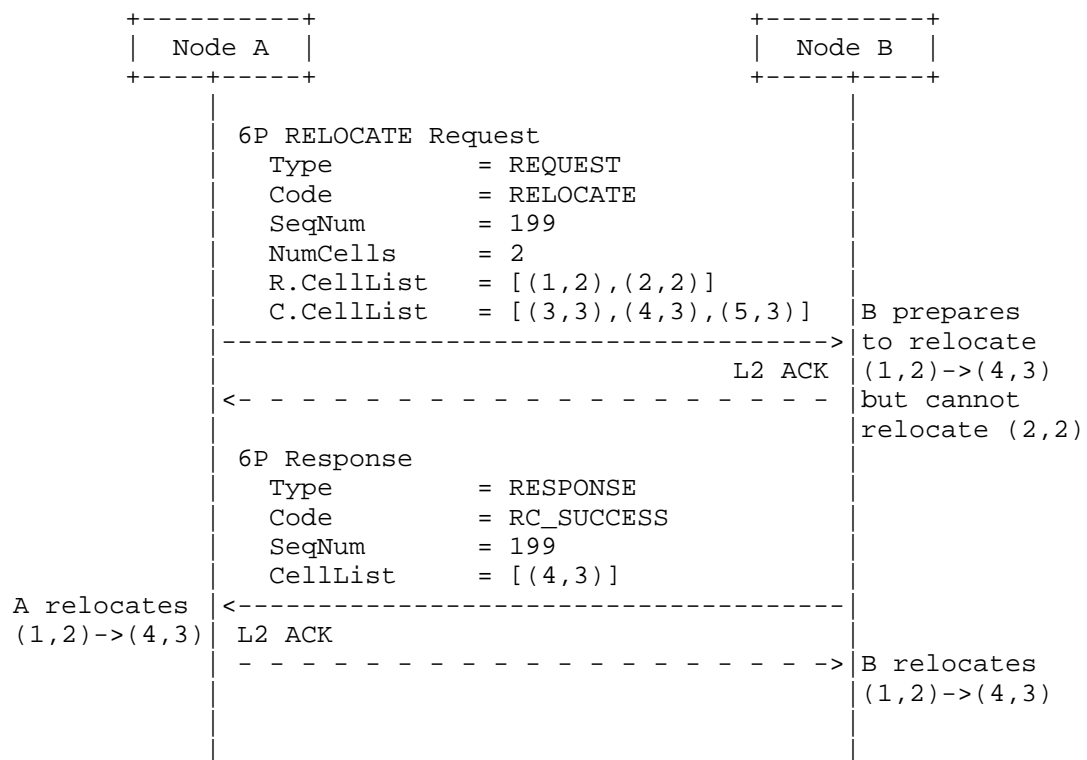


Figure 17: Example of a partially successful 2-step 6P RELOCATION Transaction.

Figure 18 shows an example of a failed 2-step 6P RELOCATION Transaction.

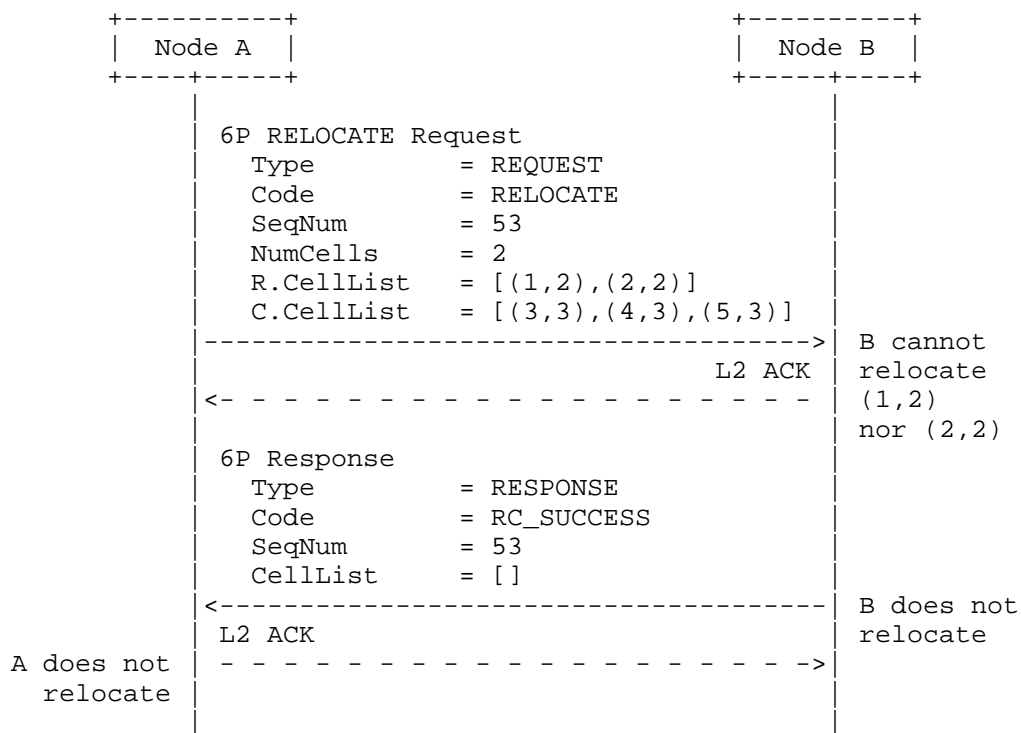


Figure 18: Failed 2-step 6P RELOCATION Transaction Example.

Figure 19 shows an example of a successful 3-step 6P RELOCATION Transaction.

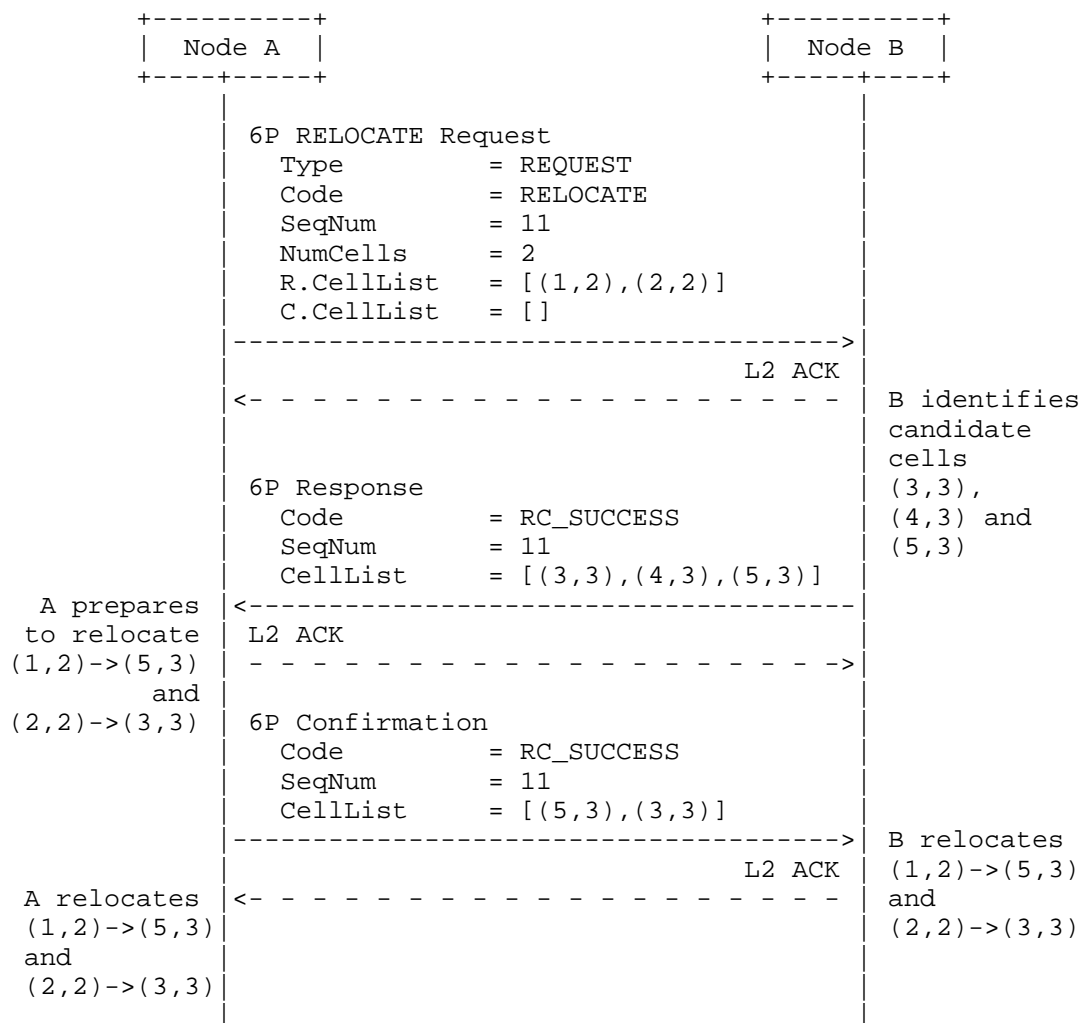


Figure 19: Example of a successful 3-step 6P RELOCATION Transaction.

3.3.4. Counting Cells

To retrieve the number of scheduled cells node A has with B, node A issues a 6P COUNT command. The Type field (T) is set to REQUEST. The Code field is set to COUNT. Figure 20 defines the format of a 6P COUNT Request.

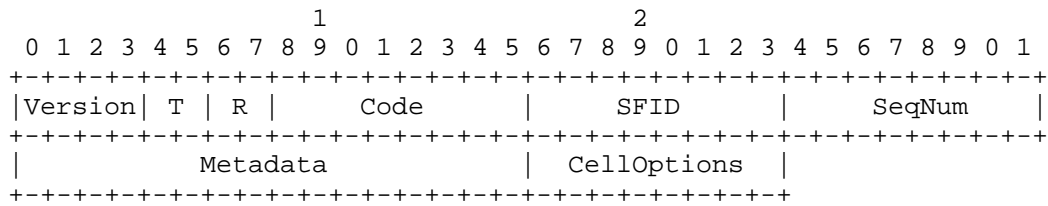


Figure 20: 6P COUNT Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1.

Its format is the same as that in the 6P ADD command, but its content could be different.

CellOptions: Specifies which type of cell to be counted.

Figure 21 defines the format of a 6P COUNT Response.

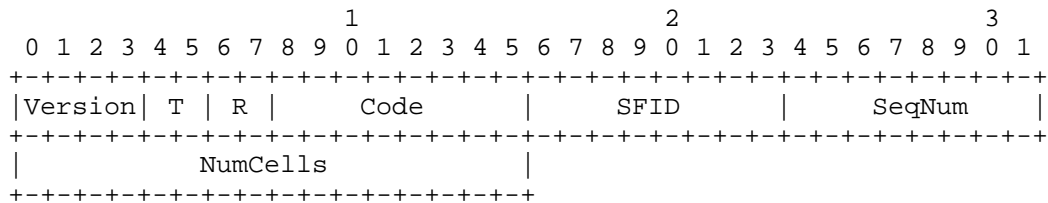


Figure 21: 6P COUNT Response Format.

NumCells: The number of cells which correspond to the fields of the request.

Node A issues a COUNT command to node B, specifying some cell options. Upon receiving the 6P COUNT request, node B goes through its schedule and counts the number of cells scheduled with node A in its own schedule which match the cell options in the CellOptions field of the request. Section 3.2.3 details the use of the CellOptions field.

Node B issues a 6P response to node A with return code set to RC_SUCCESS, and with NumCells containing the number of cells that match the request.

3.3.5. Listing Cells

To retrieve a list of scheduled cells node A has with node B, node A issues a 6P LIST command. The Type field (T) is set to REQUEST. The Code field is set to LIST. Figure 22 defines the format of a 6P LIST Request.

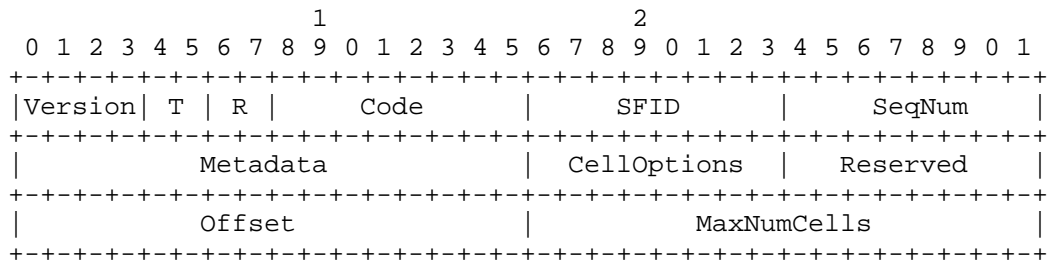


Figure 22: 6P LIST Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1.

Its format is the same as that in the 6P ADD command, but its content could be different.

CellOptions: Specifies which type of cell to be listed.

Reserved: Reserved bits. These bits SHOULD be set to zero when sending the message, and MUST be ignored upon reception.

Offset: The Offset of the first scheduled cell that is requested.

The mechanism assumes cells are ordered according to a rule defined in the SF. The rule MUST always order the cells in the same way.

MaxNumCells: The maximum number of cells to be listed. Node B MAY return fewer than MaxNumCells cells, for example if MaxNumCells cells do not fit in the frame.

Figure 23 defines the format of a 6P LIST Response.

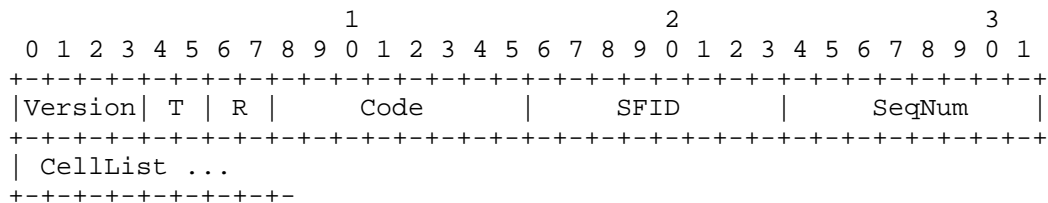


Figure 23: 6P LIST Response Format.

CellList: A list of 0 or more 6P Cells.

When receiving a LIST command, node B returns the cells scheduled with A in its schedule that match the CellOptions field as specified in Section 3.2.3.

When node B receives a LIST request, the returned CellList in the 6P Response contains between 0 and MaxNumCells cells, starting from the specified offset. Node B SHOULD include as many cells as fit in the frame. If the response contains the last cell, Node B MUST set the

Code field in the response to RC_EOL ("End of List", as per Figure 38), indicating to Node A that there no more cells that match the request. Node B MUST return at least one cell, unless the specified Offset is beyond the end of B's cell list in its schedule. If node B has fewer than Offset cells that match the request, node B returns an empty CellList and a Code field set to RC_EOL.

3.3.6. Clearing the Schedule

To clear the schedule between nodes A and B (for example after a schedule inconsistency is detected), node A issues a CLEAR command. The Type field (T) is set to 6P Request. The Code field is set to CLEAR. Figure 24 defines the format of a 6P CLEAR Request.

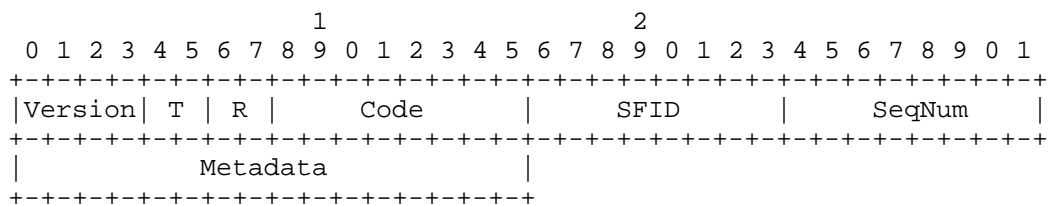


Figure 24: 6P CLEAR Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1. Its format is the same as that in the 6P ADD command, but its content could be different.

Figure 25 defines the format of a 6P CLEAR Response.

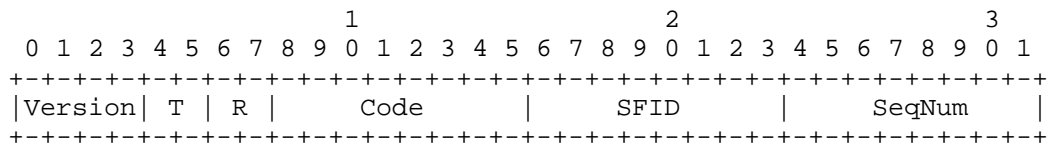


Figure 25: 6P CLEAR Response Format.

When a 6P CLEAR command is issued from node A to node B, both nodes A and B MUST remove all the cells scheduled between them. That is, node A MUST remove all the cells scheduled with node B, and node B MUST remove all the cells scheduled with node A. In a 6P CLEAR command, the SeqNum MUST NOT be checked. In particular, even if the request contains a SeqNum value that would normally cause node B to detect a schedule inconsistency, the transaction MUST NOT be aborted. Upon 6P CLEAR completion, the value of SeqNum MUST be reset to 0.

The return code to a 6P CLEAR command SHOULD be RC_SUCCESS unless the operation cannot be executed. When the CLEAR operation cannot be executed, the return code MUST be set to RC_RESET.

3.3.7. Generic Signaling Between SFs

The 6P SIGNAL message allows the SF implementations on two neighbor nodes to exchange generic commands. The payload in a received SIGNAL message is an opaque set of bytes passed unmodified to the SF. The length of the payload is determined through the length field of the Payload IE Header. How the generic SIGNAL command is used is specified by the SF, and outside the scope of this document. The Type field (T) is set to REQUEST. The Code field is set to SIGNAL. Figure 26 defines the format of a 6P SIGNAL Request.

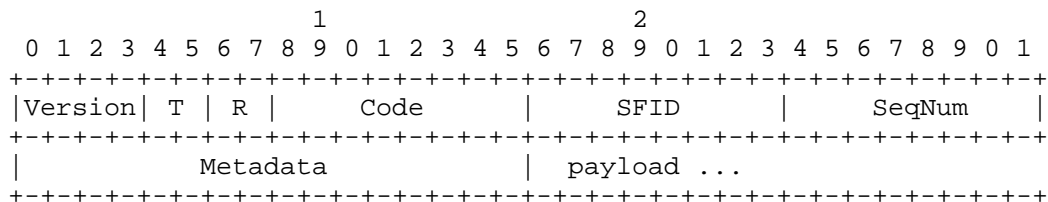


Figure 26: 6P SIGNAL Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1. Its format is the same as that in the 6P ADD command, but its content could be different.

Figure 27 defines the format of a 6P SIGNAL Response.

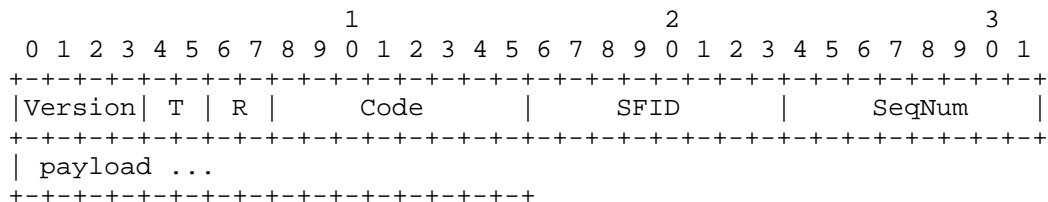


Figure 27: 6P SIGNAL Response Format.

3.4. Protocol Functional Details

3.4.1. Version Checking

All messages contain a Version field. If multiple Versions of the 6P protocol have been defined (in future specifications for Version values different from 0), a node MAY implement multiple protocol

versions at the same time. When a node receives a 6P message with a Version number it does not implement, the node MUST reply with a 6P Response with a return code field set to `RC_ERR_VERSION`. The format of this 6P Response message MUST be compliant with Version 0 and MUST be supported by all future versions of the protocol. This ensures that, when node B sends a 6P Response to node A indicating it does not implement the 6P version in the 6P Request, node A can successfully parse that response.

When a node supports a version number received in a 6P Request message, the Version field in the 6P Response MUST be the same as the Version field in the corresponding 6P Request. Similarly, in a 3-step transaction, the Version field in the 6P Confirmation MUST match that of the 6P Request and 6P Response of the same transaction.

3.4.2. SFID Checking

All messages contain an SFID field. A node MAY support multiple SFs at the same time. When receiving a 6P message with an unsupported SFID, a node MUST reply with a 6P Response with return code of `RC_ERR_SFID`. The SFID field in the 6P Response MUST be the same as the SFID field in the corresponding 6P Request. In a 3-step transaction, the SFID field in the 6P Confirmation MUST match that of the 6P Request and the 6P Response of the same transaction.

3.4.3. Concurrent 6P Transactions

Only a single 6P Transaction between two neighbors, in a given direction, can take place at the same time. That is, a node MUST NOT issue a new 6P Request to a given neighbor before the previous 6P Transaction it initiated has finished (possibly timed out). If a node receives a 6P Request from a given neighbor before having sent the 6P Response to the previous 6P Request from that neighbor, it MUST send back a 6P Response with a return code of `RC_RESET` (as per Figure 38) and discard this ongoing second transaction. A node receiving a `RC_RESET` code MUST abort the second transaction and consider it never happened (i.e. reverting changes to the schedule or SeqNum done by this transaction).

Nodes A and B MAY support having two transactions going on at the same time, one in each direction. Similarly, a node MAY support concurrent 6P Transactions with different neighbors. In this case, the cells involved in an ongoing 6P Transaction MUST be "locked" until the transaction finishes. For example, in Figure 1, node C can have a different ongoing 6P Transaction with nodes B and R. In case a node does not have enough resources to handle concurrent 6P Transactions from different neighbors it MUST reply with a 6P Response with return code `RC_ERR_BUSY` (as per Figure 38). In case

the requested cells are locked, it MUST reply to that request with a 6P Response with return code RC_ERR_LOCKED (as per Figure 38). The node receiving RC_ERR_BUSY or a RC_ERR_LOCKED MAY implement a retry mechanism, defined by the SF.

3.4.4. 6P Timeout

A timeout occurs when the node that successfully sent a 6P Request does not receive the corresponding 6P Response within an amount of time specified by the SF. In a 3-step transaction, a timeout also occurs when a node sending the 6P Response does not receive a 6P Confirmation. When a timeout occurs, the transaction MUST be canceled at the node where the timeout occurs. The value of the 6P Timeout should be larger than the longest possible time it takes to receive the 6P Response or Confirmation. The value of the 6P Timeout hence depends on the number of cells scheduled between the neighbor nodes, the maximum number of link-layer retransmissions, etc. The SF MUST determine the value of the timeout. The value of the timeout is out of scope of this document.

3.4.5. Aborting a 6P Transaction

In case the receiver of a 6P Request fails during a 6P Transaction and is unable to complete it, it SHOULD reply to that Request with a 6P Response with return code RC_RESET. Upon receiving this 6P Response, the initiator of the 6P Transaction MUST consider the 6P Transaction as failed.

Similarly, in the case of 3-step transaction, when the receiver of a 6P Response fails during the 6P Transaction and is unable to complete it, it MUST reply to that 6P Response with a 6P Confirmation with return code RC_RESET. Upon receiving this 6P Confirmation, the sender of the 6P Response MUST consider the 6P Transaction as failed.

3.4.6. SeqNum Management

The SeqNum is the field in the 6top IE header used to match Request, Response and Confirmation. The SeqNum is used to detect and handle duplicate commands (Section 3.4.6.1) and schedule inconsistencies (Section 3.4.6.2). Each node remembers the last used SeqNum for each neighbor. That is, a node stores as many SeqNum values as it has neighbors. In case of supporting multiple SFs at a time, a SeqNum value is maintained per SF and per neighbor. In the remainder of this section, we describe the use of SeqNum between two neighbors; the same happens for each other neighbor, independently.

When a node resets or after a CLEAR transaction, it MUST reset SeqNum to 0. The 6P Response and 6P Confirmation for a transaction MUST use

the same SeqNum value as that in the Request. After every transaction, the SeqNum MUST be incremented by exactly 1.

Specifically, if node A receives the link-layer acknowledgment for its 6P Request, it commits to incrementing the SeqNum by exactly 1 after the 6P Transaction ends. This ensure that, at the next 6P Transaction where it sends a 6P Request, 6P Request will have a different SeqNum.

Similarly, a node B increments the SeqNum by exactly 1 after having received the link-layer acknowledgment for the 6P Response (2-step 6P Transaction), or after having sent the link-layer acknowledgment for the 6P Confirmation (3-step 6P Transaction) .

When a node B receives a 6P Request from node A with SeqNum equal to 0, it checks the stored SeqNum for A. If A is a new neighbor, the stored SeqNum in B will be 0. The transaction can continue. If the stored SeqNum for A in B is different than 0, a potential inconsistency is detected. In this case, B MUST return RC_ERR_SEQNUM with SeqNum=0. The SF of node A MAY decide what to do next, as described in Section 3.4.6.2.

The SeqNum MUST be implemented as a lollipop counter: it rolls over from 0xFF to 0x01 (not to 0x00). This is used to detect a neighbor reset. Figure 28 lists the possible values of the SeqNum.

Value	Meaning
0x00	Clear or After device Reset
0x01-0xFF	Lollipop Counter values

Figure 28: Possible values of the SeqNum.

3.4.6.1. Detecting and Handling Duplicate 6P Messages

All 6P commands are link-layer acknowledged. A duplicate message means that a node receives a second 6P Request, Response or Confirmation. This happens when the link-layer acknowledgment is not received, and a link-layer retransmission happens. Duplicate messages are normal and unavoidable.

Figure 29 shows an example 2-step transaction in which Node A receives a duplicate 6P Response.

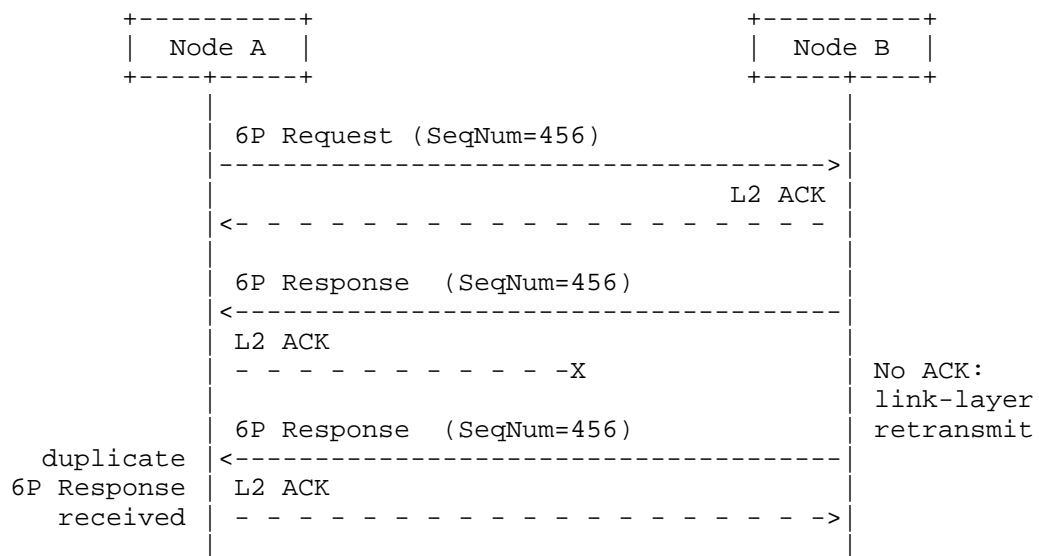


Figure 29: Example duplicate 6P message.

Figure 30 shows example 3-step transaction in which Node A receives a out-of-order duplicate 6P Response after having sent a 6P Confirmation.

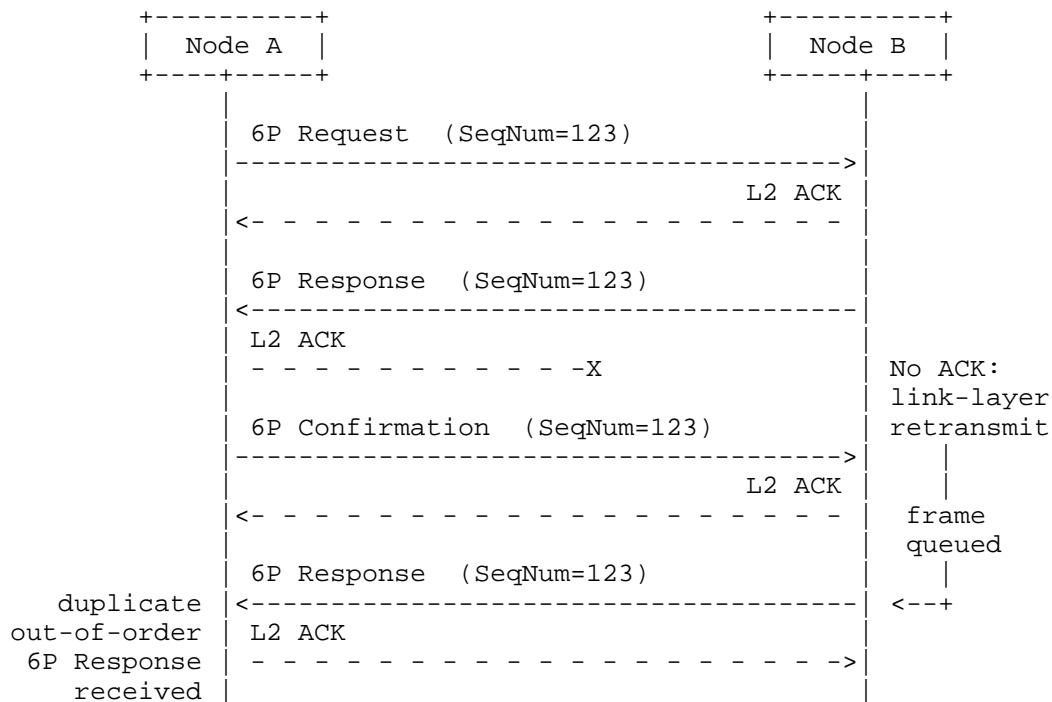


Figure 30: Example out-of-order duplicate 6P message.

A node detects a duplicate 6P message when it has the same SeqNum and type as the last frame received from the same neighbor. When receiving a duplicate 6P message, a node **MUST** send a link-layer acknowledgment, but **MUST** silently ignore the 6P message at the 6top sublayer.

3.4.6.2. Detecting and Handling a Schedule Inconsistency

A schedule inconsistency happens when the schedules of nodes A and B are inconsistent. For example, when node A has a transmit cell to node B, but node B does not have the corresponding receive cell, and therefore isn't listening to node A on that cell. A schedule inconsistency results in loss of connectivity.

The SeqNum field, which is present in each 6P message, is used to detect an inconsistency. The SeqNum field increments by 1 at each message, as detailed in Section 3.4.6. A node computes the expected SeqNum field for the next 6P Transaction. If a node receives a 6P Request with a SeqNum value that is not the expected one, it has detected an inconsistency.

There are at least 2 cases in which a schedule inconsistency happens.

The first case is when a node loses state, for example when it is power cycled (turned off, then on). In that case, its SeqNum value is reset to 0. Since the SeqNum is a lollipop counter, its neighbor detects an inconsistency at the next 6P transaction. This is illustrated in Figure 31 and Figure 32.

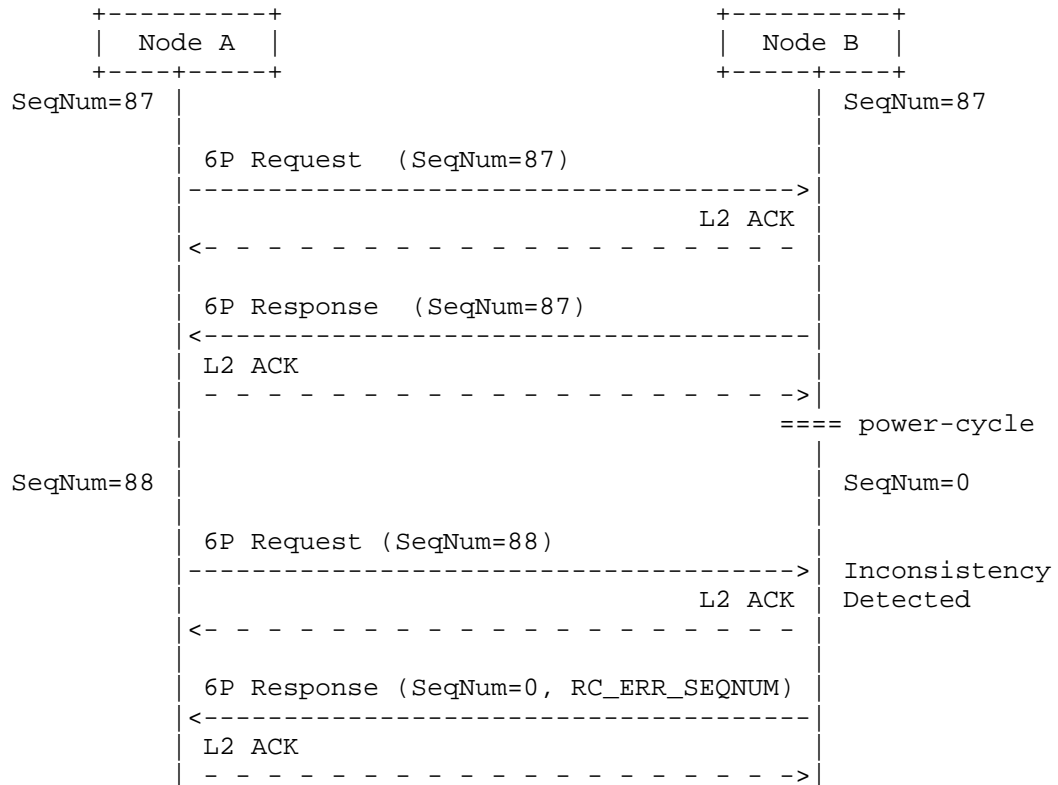


Figure 31: Example of inconsistency because of node B reset.
Detected by node B

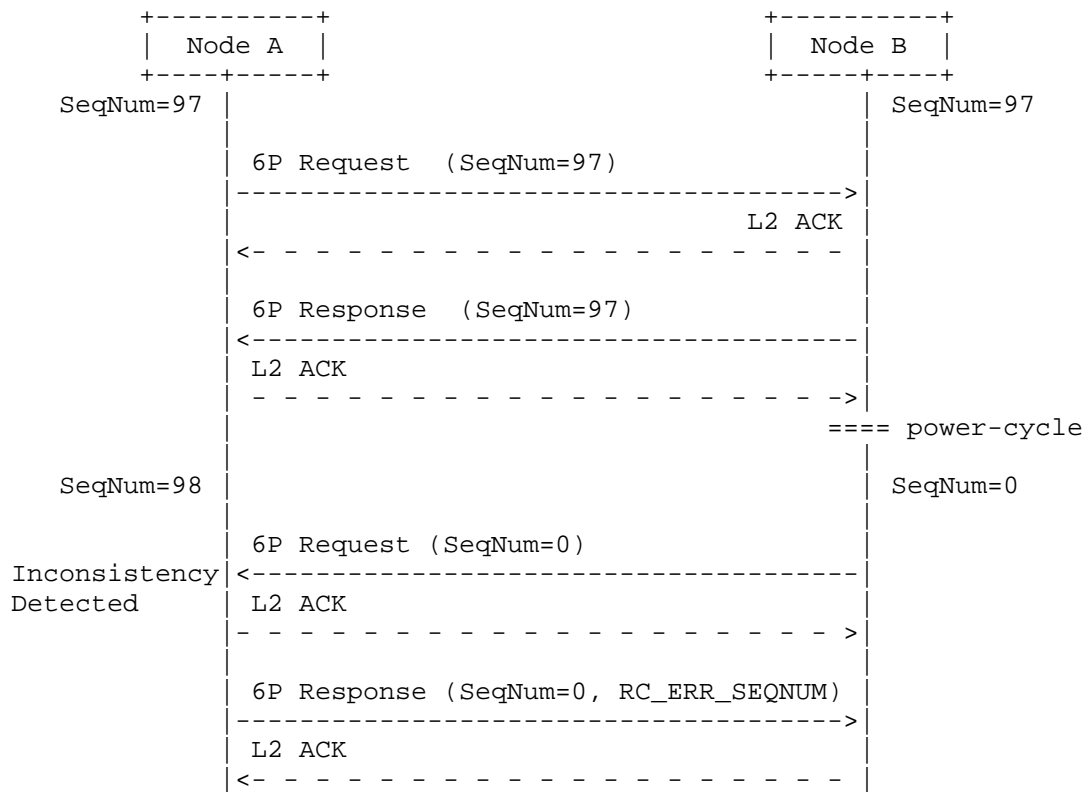


Figure 32: Example of inconsistency because node B resets. Detected by node A

The second case is when the maximum number of link-layer retransmissions is reached on the 6P Response of a 2-step transaction (or equivalently on a 6P Confirmation of a 3-step transaction). This is illustrated in Figure 33.

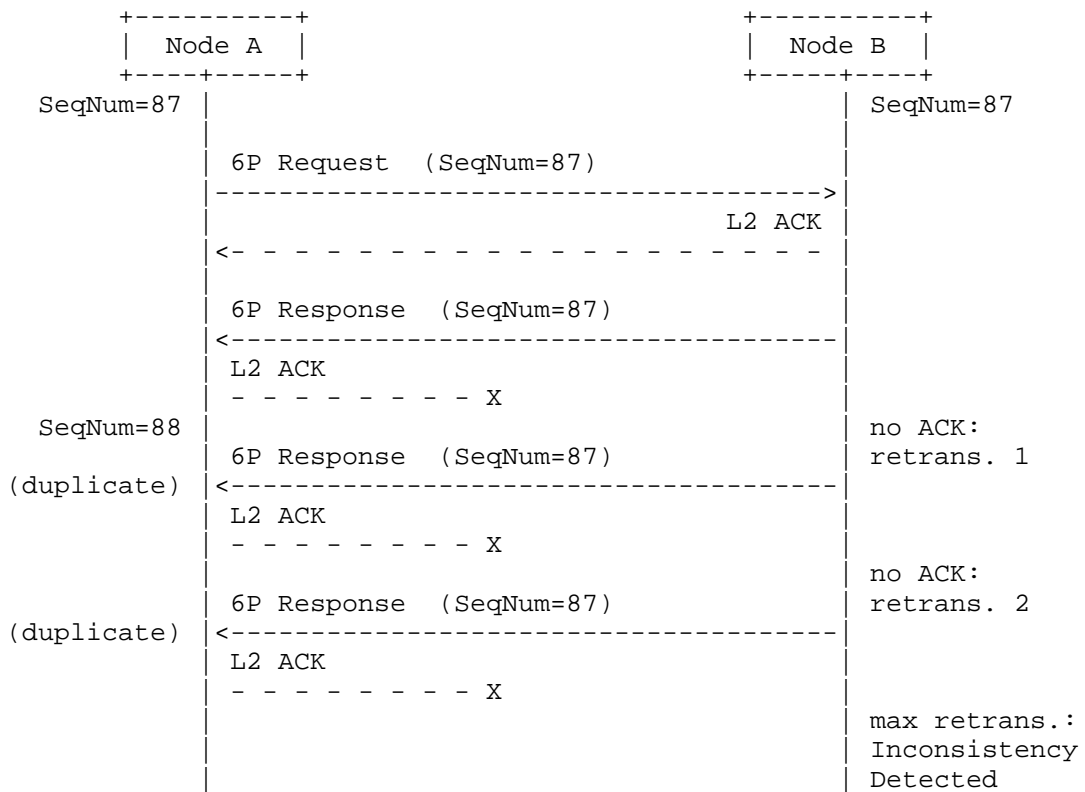


Figure 33: Example inconsistency because of maximum link-layer retransmissions (here 2).

In both cases, node B detects the inconsistency.

If the inconsistency is detected during a 6P Transaction (Figure 31), the node that has detected it MUST send back a 6P Response or 6P Confirmation with an error code of RC_ERR_SEQNUM. In this 6P Response or 6P Confirmation, the SeqNum field MUST be set to the value of the sender of the message (0 in the example in Figure 31).

The SF of the node which has detected the inconsistency MUST define how to handle the inconsistency. A first possibility is to issue a 6P CLEAR request to clear the schedule, and rebuild. A second possibility is to issue a 6P LIST request to retrieve the schedule. A third possibility is to internally "roll-back" the schedule. How to handle an inconsistency is out of scope of this document. The SF defines how to handle an inconsistency.

3.4.7. Handling Error Responses

A return code marked as Yes in the "Is Error" column in Figure 38 indicates an error. When a node receives a 6P Response or 6P Confirmation with an error, it MUST consider the 6P Transaction as failed. In particular, if this was a response to a 6P ADD, DELETE or RELOCATE Request, the node MUST NOT add, delete or relocate any of the cells involved in this 6P Transaction. Similarly, a node sending a 6P Response or a 6P Confirmation with an error code MUST NOT add, delete, relocate any cells as part of that 6P Transaction. If a node receives an unrecognized return code the 6P Transaction MUST be considered as failed. In particular, in a 3 step 6P Transaction, a 6P Response with an unrecognized return code MUST be responded with a 6P Confirmation with return code RC_ERR and consider the transaction as failed. Defining what to do after an error has occurred is out of scope of this document. The SF defines what to do after an error has occurred.

3.5. Security

6P messages MUST be secured through link-layer security. This is possible because 6P messages are carried as Payload IEs.

4. Requirements for 6top Scheduling Functions (SF) Specification

4.1. SF Identifier (SFID)

Each SF has a 1-byte identifier. Section 6.2.5 defines the rules for applying for an SFID.

4.2. Requirements for an SF specification

The specification for an SF

- o MUST specify an identifier for that SF.
- o MUST specify the rule for a node to decide when to add/delete one or more cells to a neighbor.
- o MUST specify the rule for a Transaction source to select cells to add to the CellList field in the 6P ADD Request.
- o MUST specify the rule for a Transaction destination to select cells from CellList to add to its schedule.
- o MUST specify a value for the 6P Timeout, or a rule/equation to calculate it.
- o MUST specify the rule for ordering cells.
- o MUST specify a meaning for the "Metadata" field in the 6P ADD Request.
- o MUST specify the SF behavior of a node when it boots.
- o MUST specify how to handle a schedule inconsistency.

- o MUST specify what to do after an error has occurred (either the node sent a 6P Response with an error code, or received one).
- o MUST specify the list of statistics to gather. Example statistics include the number of transmitted frames to each neighbor. In case the SF requires no statistics to be gathered, the specific of the SF MUST explicitly state so.
- o SHOULD clearly state the application domain the SF is created for.
- o SHOULD contain examples which highlight normal and error scenarios.
- o SHOULD contain a list of current implementations, at least during the I-D state of the document, per [RFC6982].
- o SHOULD contain a performance evaluation of the scheme, possibly through references to external documents.
- o SHOULD define the format of the SIGNAL command payload and its use.
- o MAY redefine the format of the CellList field.
- o MAY redefine the format of the CellOptions field.
- o MAY redefine the meaning of the CellOptions field.

5. Security Considerations

6P messages are carried inside 802.15.4 Payload Information Elements (IEs). Those Payload IEs are encrypted and authenticated at the link layer through CCM* [CCM-Star]. 6P benefits from the same level of security as any other Payload IE. The 6P protocol does not define its own security mechanisms. In particular, although a key management solution is out of scope of this document, the 6P protocol will benefit for the key management solution used in the network. This is relevant as security attacks such as forgery and misattribution attacks become more damaging when a single key is shared amongst a group of more than 2 participants.

The 6P protocol does not provide protection against DOS attacks. Example attacks include, not sending confirmation messages in 3-step transaction, and sending wrongly formatted requests. These cases SHOULD be handled by an appropriate policy, such as rate-limiting or time-limited blacklisting the attacker after several attempts. The effect on the overall network is mostly localized to those two nodes, as communication happens in dedicated cells.

6. IANA Considerations

6.1. IETF IE Subtype '6P'

This document adds the following number to the "IEEE Std 802.15.4 IETF IE subtype IDs" registry defined by [RFC8137]:

Value	Subtype ID	Reference
<TBD>	SUBID_6TOP	RFCXXXX

Figure 34: IETF IE Subtype SUBID_6TOP.

6.2. 6TiSCH parameters sub-registries

This section defines sub-registries within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry, hereafter referred to as the "6TiSCH parameters" registry. Each sub-registry is described in a subsection.

6.2.1. 6P Version Numbers

The name of the sub-registry is "6P Version Numbers".

A Note included in this registry should say: "In the 6top Protocol (6P) [RFCXXXX] there is a field to identify the version of the protocol. This field is 4 bits in size."

Each entry in the sub-registry must include the Version in the range 0-15, and a reference to the 6P version's documentation.

The initial entry in this sub-registry is as follows:

Version	Reference
0	RFCXXXX

Figure 35: 6P Version Numbers.

All other Version Numbers are Unassigned.

The IANA policy for future additions to this sub-registry is "IETF Review or IESG Approval" as described in [RFC8126].

6.2.2. 6P Message Types

The name of the sub-registry is "6P Message Types".

A note included in this registry should say: "In the 6top Protocol (6P) version 0 [RFCXXXX], there is a field to identify the type of message. This field is 2 bits in size."

Each entry in the sub-registry must include the Type in range b00-b11, the corresponding Name, and a reference to the 6P message type's documentation.

Initial entries in this sub-registry are as follows:

Type	Name	Reference
b00	REQUEST	RFCXXXX
b01	RESPONSE	RFCXXXX
b10	CONFIRMATION	RFCXXXX

Figure 36: 6P Message Types.

All other Message Types are Reserved.

The IANA policy for future additions to this sub-registry is "IETF Review or IESG Approval" as described in [RFC8126].

6.2.3. 6P Command Identifiers

The name of the sub-registry is "6P Command Identifiers".

A Note included in this registry should say: "In the 6top Protocol (6P) version 0 [RFCXXXX], there is a Code field which is 8 bits in size. In a 6P Request, the value of this Code field is used to identify the command."

Each entry in the sub-registry must include an Identifier in the range 0-255, the corresponding Name, and a reference to the 6P command identifier's documentation.

Initial entries in this sub-registry are as follows:

Identifier	Name	Reference
0	Reserved	
1	ADD	RFCXXXX
2	DELETE	RFCXXXX
3	RELOCATE	RFCXXXX
4	COUNT	RFCXXXX
5	LIST	RFCXXXX
6	SIGNAL	RFCXXXX
7	CLEAR	RFCXXXX
8-254	Unassigned	
255	Reserved	

Figure 37: 6P Command Identifiers.

The IANA policy for future additions to this sub-registry is "IETF Review or IESG Approval" as described in [RFC8126].

6.2.4. 6P Return Codes

The name of the sub-registry is "6P Return Codes".

A Note included in this registry should say: "In the 6top Protocol (6P) version 0 [RFCXXXX], there is a Code field which is 8 bits in size. In a 6P Response or 6P Confirmation, the value of this Code field is used to identify the return code."

Each entry in the sub-registry must include a Code in the range 0-255, the corresponding Name, the corresponding Description, and a reference to the 6P return code's documentation.

Initial entries in this sub-registry are as follows:

Code	Name	Description	Is Error?
0	RC_SUCCESS	operation succeeded	No
1	RC_EOL	end of list	No
2	RC_ERR	generic error	Yes
3	RC_RESET	critical error, reset	Yes
4	RC_ERR_VERSION	unsupported 6P version	Yes
5	RC_ERR_SFID	unsupported SFID	Yes
6	RC_ERR_SEQNUM	schedule inconsistency	Yes
7	RC_ERR_CELLLIST	cellList error	Yes
8	RC_ERR_BUSY	busy	Yes
9	RC_ERR_LOCKED	cells are locked	Yes

Figure 38: 6P Return Codes.

All other Message Types are Unassigned.

The IANA policy for future additions to this sub-registry is "IETF Review or IESG Approval" as described in [RFC8126].

6.2.5. 6P Scheduling Function Identifiers

6P Scheduling Function Identifiers.

A Note included in this registry should say: "In the 6top Protocol (6P) version 0 [RFCXXXX], there is a field to identify the scheduling function to handle the message. This field is 8 bits in size."

Each entry in the sub-registry must include an SFID in the range 0-255, the corresponding Name, and a reference to the 6P Scheduling Function's documentation.

Initial entries in this sub-registry are as follows:

SFID	Name	Reference
0	Minimal Scheduling Function (MSF)	draft-chang-6tisch-msf

Figure 39: SF Identifiers (SFID).

All other Message Types are Unassigned.

The IANA policy for future additions to this sub-registry depends on the value of the SFID, as defined in Figure 40. These specifications must follow the guidelines of Section 4.

Range	Registration Procedures
0-127	IETF Review or IESG Approval
128-255	Expert Review

Figure 40: SF Identifier (SFID): Registration Procedure.

6.2.6. 6P CellOptions bitmap

The name of the sub-registry is "6P CellOptions bitmap".

A Note included in this registry should say: "In the 6top Protocol (6P) version 0 [RFCXXXX], there is an optional CellOptions field which is 8 bits in size."

Each entry in the sub-registry must include a bit position in the range 0-7, the corresponding Name, and a reference to the bit's documentation.

Initial entries in this sub-registry are as follows:

bit	Name	Reference
0	TX (Transmit)	RFCXXXX
1	RX (Receive)	RFCXXXX
2	SHARED	RFCXXXX
3-7	Reserved	

Figure 41: 6P CellOptions bitmap.

All other Message Types are Reserved.

The IANA policy for future additions to this sub-registry is "IETF Review or IESG Approval" as described in [RFC8126].

7. References

7.1. Normative References

- [IEEE802154]
IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 - IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", October 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8137] Kivinen, T. and P. Kinney, "IEEE 802.15.4 Information Element for the IETF", RFC 8137, DOI 10.17487/RFC8137, May 2017, <<https://www.rfc-editor.org/info/rfc8137>>.

7.2. Informative References

- [CCM-Star]
Struik, R., "Formal Specification of the CCM* Mode of Operation, IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs).", September 2005.
- [OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.
- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<https://www.rfc-editor.org/info/rfc6982>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<https://www.rfc-editor.org/info/rfc7554>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

[RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <<https://www.rfc-editor.org/info/rfc8180>>.

Appendix A. Recommended Structure of an SF Specification

The following section structure for a SF document is RECOMMENDED:

- o Introduction
- o Scheduling Function Identifier
- o Rules for Adding/Deleting Cells
- o Rules for CellList
- o 6P Timeout Value
- o Rule for Ordering Cells
- o Meaning of the Metadata Field
- o Node Behavior at Boot
- o Schedule Inconsistency Handling
- o 6P Error Handling
- o Examples
- o Implementation Status
- o Security Considerations
- o IANA Considerations

Authors' Addresses

Qin Wang (editor)
Univ. of Sci. and Tech. Beijing
30 Xueyuan Road
Beijing, Hebei 100083
China

Email: wangqin@ies.ustb.edu.cn

Xavier Vilajosana
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Email: xvilajosana@uoc.edu

Thomas Watteyne
Analog Devices
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: thomas.watteyne@analog.com

6TiSCH
Internet-Draft
Intended status: Experimental
Expires: January 3, 2018

D. Dujovne, Ed.
Universidad Diego Portales
LA. Grieco
Politecnico di Bari
MR. Palattella
Luxembourg Institute of Science and Technology (LIST)
N. Accettura
LAAS-CNRS
July 2, 2017

6TiSCH 6top Scheduling Function Zero (SF0)
draft-ietf-6tisch-6top-sf0-05

Abstract

This document defines a Scheduling Function called "Scheduling Function Zero" (SF0). SF0 dynamically adapts the number of scheduled cells between neighbor nodes, based on the amount of currently allocated cells and the neighbor nodes' cell requirements. Neighbor nodes negotiate in a distributed neighbor-to-neighbor basis the number of cell(s) to be added/deleted. SF0 uses the 6P signaling messages to add/delete cells in the schedule. This function selects the candidate cells from the schedule, defines which cells will be added/deleted and triggers the allocation/deallocation process.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 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. TEMPORARY EDITORIAL NOTES	3
2. Introduction	3
3. Scheduling Function Identifier	4
4. Allocated and Used Cells	4
5. Overprovisioning	4
6. Scheduling Algorithm	4
6.1. SF0 Triggering Events	4
6.2. SF0 Cell Estimation Algorithm	4
6.3. SF0 Allocation Policy	6
7. Rules for CellList	8
8. 6P Timeout Value	8
9. Meaning of Metadata Information	9
10. Node Behavior at Boot	9
11. Cell Type	9
12. SF0 Statistics	9
13. Relocating Cells	9
14. Forced Cell Deletion Policy	10
15. 6P Error Handling	10
16. Examples	10
17. Implementation Status	10
18. Security Considerations	11
19. IANA Considerations	11
20. 6P Compliance	11
21. Acknowledgments	12
22. References	12
22.1. Normative References	12
22.2. Informative References	12
Appendix A. [TEMPORARY] Changelog	13
Authors' Addresses	13

1. TEMPORARY EDITORIAL NOTES

This document is an Internet Draft, so it is work-in-progress by nature. It contains the following work-in-progress elements:

- o "TODO" statements are elements which have not yet been written by the authors for some reason (lack of time, ongoing discussions with no clear consensus, etc). The statement does indicate that the text will be written at some time.
- o "TEMPORARY" appendices are there to capture current ongoing discussions, or the changelog of the document. These appendices will be removed in the final text.
- o "IANA_" identifiers are placeholders for numbers assigned by IANA. These placeholders are to be replaced by the actual values they represent after their assignment by IANA.
- o The string "REMARK" is put before a remark (questions, suggestion, etc) from an author, editor or contributor. These are on-going discussions at the time of writing, NOT part of the final text.
- o This section will be removed in the final text.

2. Introduction

This document defines a minimal Scheduling Function using the 6P protocol [I-D.ietf-6tisch-6top-protocol], called "Scheduling Function Zero" (SF0). SF0 is designed to offer a number of functionalities to be usable in a wide range of applications. SF0 defines two algorithms: The Scheduling Algorithm defines the number of cells to allocate/delete between two neighbours and the Relocation Algorithm defines when to relocate a cell.

To synthesize, a node running SF0 determines when to add/delete cells in a three-step process:

1. It waits for a triggering event (Section 6.1).
2. It applies the Cell Estimation Algorithm (CEA) for a particular neighbor to determine how many cells are required to that neighbor (Section 6.2).
3. It applies the Allocation Policy to compare the number of required cells to the number of already scheduled cells, and determines the number of cells to add/delete (Section 6.3).

We expect additional SFs, offering more functionalities for a more specific use case, to be defined in future documents. SF0 addresses the requirements for a scheduling function listed in Section 5.2 from [I-D.ietf-6tisch-6top-protocol], and follows the recommended outline listed in Section 5.3 of [I-D.ietf-6tisch-6top-protocol]. This document follows the terminology defined in [I-D.ietf-6tisch-terminology].

3. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of SF0 is IANA_6TISCH_SFID_SF0.

4. Allocated and Used Cells

An allocated cell is assigned as a TX, RX or Shared cell on the schedule, as a reserved resource. This reservation does not imply that a packet will be transmitted during the scheduled cell time. A used cell is a cell where a packet has been transmitted during the scheduled cell time on the last slotframe.

5. Overprovisioning

Overprovisioning is the action and effect of increasing a value representing an amount of resources. In the case of SF0, overprovisioning is done as a provision to reduce traffic variability effects on packet loss, to the expense of artificially allocating a number of cells.

6. Scheduling Algorithm

A number of TX cells must be allocated between neighbor nodes in order to enable data transmission among them. A portion of these allocated cells will be used by neighbors, while the remaining cells can be over-provisioned to handle unanticipated increases in cell requirements. The Scheduling Algorithm collects the cell allocation/deallocation requests from the neighbors and the number of cells which are currently under usage. First, the Cell Estimation Algorithm calculates the number of required cells and second, the calculated number is transferred to the Allocation Policy. In order to reduce consumption, this algorithm is triggered only when there is a change on the number of used cells from a particular node.

6.1. SF0 Triggering Events

We RECOMMEND SF0 to be triggered at by the following event: If there is a change on the number of used cells towards any of the neighbours. The exact mechanism of when SF0 is triggered is implementation-specific.

6.2. SF0 Cell Estimation Algorithm

The Cell Estimation Algorithm takes into account the number of current used cells to the neighbour. This allows the algorithm to estimate a new number of cells to be scheduled to the neighbour. As

a consequence, the Cell Estimation Algorithm for SF0 follows the steps described below:

1. Collect the current number of used cells to the neighbour
2. Calculate the new number of cells to be scheduled to the neighbour by adding the current number of used cells plus an OVERPROVISION number of cells
3. Transfer the request to the allocation policy as REQUIREDCELLS
4. Return to step 1 and wait for a triggering event.

The Cell Estimation Algorithm is depicted on figure Figure 1. The OVERPROVISION parameter is calculated as a percentage of the number of currently scheduled cells to the neighbour. OVERPROVISION is added to the amount of used cells to the neighbour to reduce the probability of packet loss given a sudden growth on the number of used cells to the neighbour. The OVERPROVISION value is implementation-specific. A value of OVERPROVISION equal to zero leads to queue growth and possible packet loss: In this case, there are no overprovisioned cells where a sudden growth on the number of cells can be absorbed and detected.

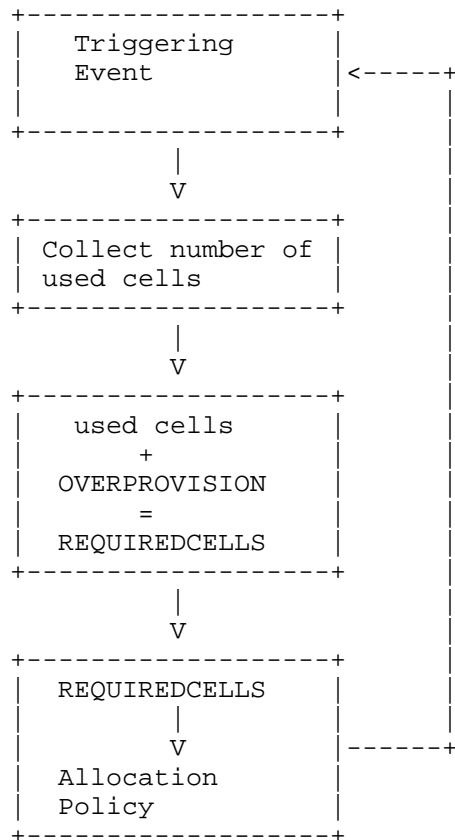


Figure 1: The SF0 Estimation Algorithm

6.3. SF0 Allocation Policy

The "Allocation Policy" is the set of rules used by SF0 to decide when to add/delete cells to a particular neighbor to satisfy the cell requirements.

SF0 uses the following parameters:

- SCHEDULEDCELLS: The number of cells scheduled from the current node to a particular neighbor.
- REQUIREDCELLS: The number of cells calculated by the Cell Estimation Algorithm from the current node to that neighbor.
- SF0THRESH: Threshold parameter introducing cell over-provisioning in the allocation policy. It is a non-negative value expressed as number of cells. The definition of this value is implementation-

specific. A setting of `SF0THRESH>0` will cause the node to allocate at least `SF0THRESH` cells to each of its' neighbors.

The SF0 allocation policy compares `REQUIREDCELLS` with `SCHEDULEDCELLS` and decides to add/delete cells taking into account `SF0THRESH`. This is illustrated in Figure 2. The number of cells to be added/deleted is out of the scope of this document and it is implementation-dependent.

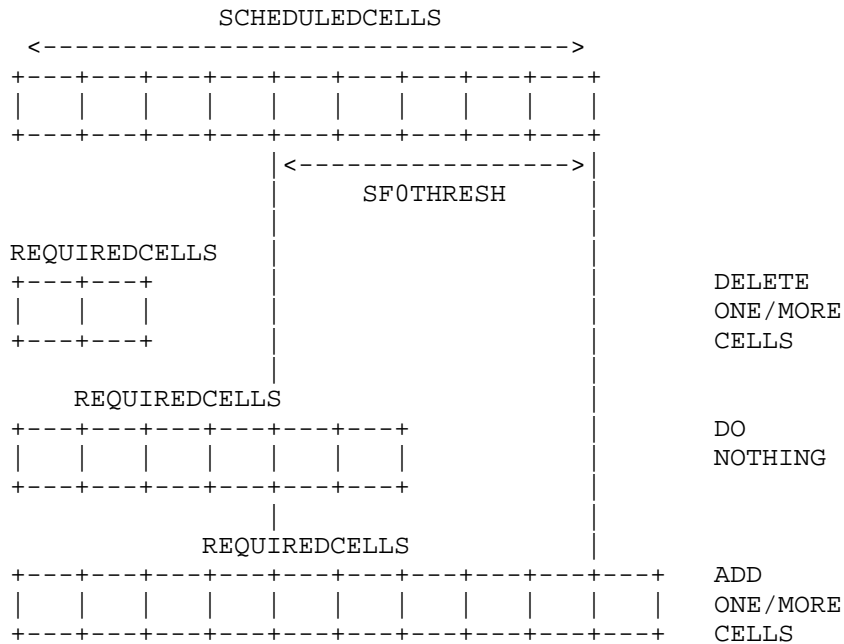


Figure 2: The SF0 Allocation Policy

1. If $REQUIREDCELLS < (SCHEDULEDCELLS - SF0THRESH)$, delete one or more cells.
2. If $(SCHEDULEDCELLS - SF0THRESH) \leq REQUIREDCELLS \leq SCHEDULEDCELLS$, do nothing.
3. If $SCHEDULEDCELLS < REQUIREDCELLS$, add one or more cells.

When `SF0THRESH` equals 0, any discrepancy between `REQUIREDCELLS` and `SCHEDULEDCELLS` triggers an action to add/delete cells. Positive values of `SF0THRESH` reduce the number of 6P Transactions. The number of cells to add or delete is implementation-specific.

7. Rules for CellList

There are two methods to define the CellList: The Whitelist method, which fills the CellList with the number of proposed cells to the neighbour, and the Blacklist, which fills the CellList with the cells which cannot be used by the neighbour. The rule to select the method is implementation-specific. When issuing a 6top ADD Request, SF0 executes the following sequence:

Whitelist case:

The Transaction Source node: Prepares the CellList field by selecting randomly the required cells, verifying that the slot offset is not occupied and choose channelOffset randomly for each cell.

The Transaction Destination node: Goes through the cells in the CellList in order, verifying whether there are no slotOffset conflicts.

Blacklist case:

The Transaction Source node: Prepares the CellList field by building a list of currently scheduled cells into the CellList. The Transaction Destination node: Selects randomly the required cells from the unallocated cells on the schedule, verifying that the slot offset is not occupied from the ones on the CellList.

SF0 does not include any transaction retry process. If the transaction is not successful, SF0 will be retriggered on the next slotframe if the number of used cells changes.

8. 6P Timeout Value

The timeout value is implementation-specific. The timeout value MAY be different for each transaction and each neighbour. The timeout range is from 0 to 128. The timeout MUST be added as an 7-bit on the Metadata header to the neighbour. There is no measurement unit associated to the timeout value. If the timeout expires, the node issues a RESET return code will be issued to the neighbour. SF0 has no retry policy. Timeout examples are depicted on Figure 3 and Figure 4.

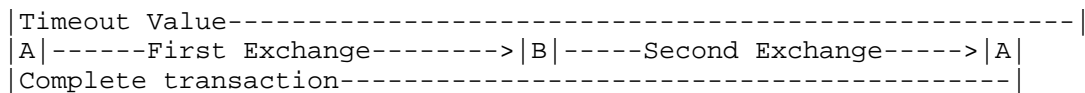


Figure 3: Example Transaction where the timeout does not expire

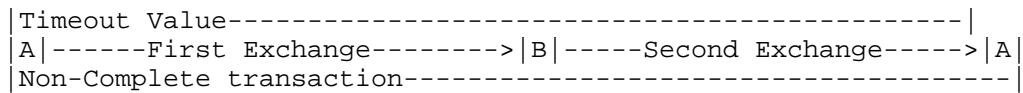


Figure 4: Example Transaction where the timeout expires

9. Meaning of Metadata Information

The Metadata 16-bit field is used as follows:

BITS 0-7 [SLOTFRAME] are used to identify the slotframe number
 BITS 8-14 [TIMEOUT] represents the Timeout value
 BIT 15 [WBLIST] is used to indicate that the CellList provided is
 a Whitelist (value=0) or a Blacklist (value=1).

10. Node Behavior at Boot

In order to define a known state after the node is restarted, a CLEAR command is issued to each of the neighbor nodes to enable a new allocation process and at least a SF0THRESH number of cells MUST be allocated to each of the neighbours.

11. Cell Type

SF0 uses TX (Transmission) cell type only, thus defining celloptions as TX=0, RX=1 and S=0 according to section 4.2.6 of [I-D.ietf-6tisch-6top-protocol].

12. SF0 Statistics

Packet Delivery Rate (PDR) is calculated per cell, as the percentage of acknowledged packets, for the last 10 packet transmission attempts. There is no retransmission policy on SF0.

13. Relocating Cells

Allocated cells may experience packet loss from different sources, such as noise, interference or cell collision (after the same cell is allocated by other nodes in range on the network).

SF0 uses Packet Delivery Rate (PDR) statistics to monitor the currently allocated cells for cell relocation (by changing their slotOffset and/or channelOffset). When the PDR of one or more softcells is below PDR_THRESHOLD, SF0 relocates each of the cell(s) to a number of available cells selected randomly. PDR_THRESHOLD is out of the scope of this document and it is implementation-dependent.

14. Forced Cell Deletion Policy

When all the cells are scheduled, we need a policy to free cells, for example, under alarm conditions or if a node disappears from the neighbor list. The action to follow this condition is out of scope of this document and it is implementation-dependent.

15. 6P Error Handling

A node implementing SF0 handles a 6P Response depending on the Return Code it contains:

RC_SUCCESS:

If the number of elements in the CellList is the number of cells specified in the NumCells field of the 6P ADD Request, the operation is complete. The node does not take further action. If the number of elements in the CellList is smaller (possibly 0) than the number of cells specified in the NumCells field of the 6P ADD Request, the neighbor has received the request, but less than NumCells of the cells in the CellList were allocated. In that case, the node MAY retry immediately with a different CellList if the amount of storage space permits, or build a new (random) CellList.

RC_ERR_VER: The node MUST NOT retry immediately. The node MAY add the neighbor node to a blacklist. The node MAY retry to contact this neighbor later.

RC_ERR_SFID: The node MUST NOT retry immediately. The node MAY add the neighbor node to a blacklist. The node MAY retry to contact this neighbor later.

RC_ERR_GEN: The node MUST issue a CLEAR command to the neighbour.

RC_ERR_BUSY: Wait for a timeout and restart the scheduling process.

RC_ERR_NORES: Wait for a timeout and restart the scheduling process.

RC_ERR_RESET: Abort 6P Transaction

RC_ERR: Abort 6P Transaction. The node MAY retry to contact this neighbor later.

16. Examples

TODO

17. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation

here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

OpenWSN: This specification is implemented in the OpenWSN project [OpenWSN]. The authors of this document are collaborating with the OpenWSN community to gather feedback about the status and performance of the protocols described in this document. Results from that discussion will appear in this section in future revision of this specification.

18. Security Considerations

TODO

19. IANA Considerations

- o IANA_6TiSCH_SFID_SF0

20. 6P Compliance

- o MUST specify an identifier for that SF. OK
- o MUST specify the rule for a node to decide when to add/delete one or more cells to a neighbor. OK
- o MUST specify the rule for a Transaction source to select cells to add to the CellList field in the 6P ADD Request. OK
- o MUST specify the rule for a Transaction destination to select cells from CellList to add to its schedule. OK
- o MUST specify a value for the 6P Timeout, or a rule/equation to calculate it. OK
- o MUST specify a meaning for the "Metadata" field in the 6P ADD Request. OK
- o MUST specify the behavior of a node when it boots. OK
- o MUST specify what to do after an error has occurred (either the node sent a 6P Response with an error code, or received one). OK
- o MUST specify the list of statistics to gather. An example statistic is the number of transmitted frames to each neighbor.

- In case the SF requires no statistics to be gathered, the specific of the SF MUST explicitly state so. OK
- o SHOULD clearly state the application domain the SF is created for. OK
 - o SHOULD contain examples which highlight normal and error scenarios.
 - o SHOULD contain a list of current implementations, at least during the I-D state of the document, per [RFC6982].
 - o SHOULD contain a performance evaluation of the scheme, possibly through references to external documents.
 - o MAY redefine the format of the CellList? field. OK

21. Acknowledgments

Thanks to Kris Pister for his contribution in designing the default Bandwidth Estimation Algorithm. Thanks to Qin Wang and Thomas Watteyne for their support in defining the interaction between SF0 and the 6top sublayer.

This work is partially supported by the Fondecyt 1121475 Project, the Inria-Chile "Network Design" group, and the IoT6 European Project (STREP) of the 7th Framework Program (Grant 288445).

22. References

22.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

22.2. Informative References

- [I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-07 (work in progress), June 2017.
- [I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-09 (work in progress), June 2017.

[OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <<http://www.rfc-editor.org/info/rfc6982>>.

Appendix A. [TEMPORARY] Changelog

- o draft-ietf-6tisch-6top-sf0-02
 - * Editorial changes (figs, typos, ...)
- o draft-ietf-6tisch-6top-sf0-03
 - * Fixed typos
 - * Removed references to "effectively used cells"
 - * Changed Cell Estimation Algorithm to the third proposed alternative on IETF97
 - * Forced cell deletion becomes implementation specific
 - * Added PDR calculation formula
 - * Added PDR_THRESHOLD as implementation specific value

Authors' Addresses

Diego Dujovne (editor)
Universidad Diego Portales
Escuela de Informatica y Telecomunicaciones
Av. Ejercito 441
Santiago, Region Metropolitana
Chile

Phone: +56 (2) 676-8121
Email: diego.dujovne@mail.udp.cl

Luigi Alfredo Grieco
Politecnico di Bari
Department of Electrical and Information Engineering
Via Orabona 4
Bari 70125
Italy

Phone: 00390805963911
Email: a.grieco@poliba.it

Maria Rita Palattella
Luxembourg Institute of Science and Technology (LIST)
Department 'Environmental Research and Innovation' (ERIN)
41, rue du Brill
Belvaux L-4422
Grand-duchy of Luxembourg

Phone: +352 275 888-5055
Email: mariarita.palattella@list.lu

Nicola Accettura
LAAS-CNRS
7, avenue du Colonel Roche
Toulouse 31400
France

Phone: +33 5 61 33 69 76
Email: nicola.accettura@laas.fr

6tisch Working Group
Internet-Draft
Intended status: Informational
Expires: August 29, 2017

M. Richardson
Sandelman Software Works
February 25, 2017

6tisch Secure Join protocol
draft-ietf-6tisch-dtsecurity-secure-join-01

Abstract

This document describes a zero-touch mechanism to enroll a new device (the "pledge") into a IEEE802.15.4 TSCH network using the 6tisch signaling mechanisms. The resulting device will obtain a domain specific credential that can be used with either 802.15.9 per-host pair keying protocols, or to obtain the network-wide key from a coordinator. The mechanism describe her is an augmentation to the one-touch mechanism described in [I-D.ietf-6tisch-minimal-security].

Status of This Memo

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

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

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

This Internet-Draft will expire on August 29, 2017.

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	2
1.1. Terminology	3
1.2. Credentials	4
1.2.1. One-Touch Assumptions	4
1.2.2. Factory provided credentials (if any)	4
1.2.3. Credentials to be introduced	5
1.3. Network Assumptions	5
1.3.1. Security above and below IP	5
1.3.2. Join network assumptions	6
1.3.3. Number and cost of round trips	6
1.3.4. Size of packets, number of fragments	7
1.4. Target end-state for join process	7
2. Join Protocol	7
2.1. Key Agreement process	8
2.2. Provisional Enrollment process	8
2.3. Key Distribution Process	9
3. YANG model for BRSKI objects	9
3.1. Description of Pledge States in Join Process	10
4. Definition of managed objects for zero-touch bootstrap	10
5. Privacy Considerations	11
5.1. Privacy Considerations for Production network	11
5.2. Privacy Considerations for New Pledges	11
5.2.1. EUI-64 derived address for join time IID	12
5.3. Privacy Considerations for Join Assistant	12
6. Security Considerations	12
7. IANA Considerations	12
8. Protocol Definition	12
9. Acknowledgements	12
10. References	12
10.1. Normative References	12
10.2. Informative References	15
10.3. URIs	16
Appendix A. appendix	16
Author's Address	16

1. Introduction

Enrollment of new nodes into LLNs present unique challenges. The constrained nodes has no user interfaces, and even if they did, configuring thousands of such nodes manually is undesirable from a human resources issue, as well as the difficulty in getting consistent results.

This document is about a standard way to introduce new nodes into a 6tisch network that does not involve any direct manipulation of the nodes themselves. This act has been called "zero-touch" provisioning, and it does not occur by chance, but requires coordination between the manufacturer of the node, the service operator running the LLN, and the installers actually taking the devices out of the shipping boxes.

The act of doing "one-touch" provisioning, where a node undergoes a site-specific indoctrination process is described in [I-D.ietf-6tisch-minimal-security].

The mechanism described here and in [I-D.ietf-6tisch-minimal-security] can be discovered by a new node in a running network, so a device which has received a network-specific "one-touch" setup, but which is located in another network, and is capable of "zero-touch" operation could discover this fact and operate in other mode.

Many of the components of the zero-touch mechanisms described here are in common with [I-D.ietf-anima-bootstrapping-keyinfra] and [I-D.ietf-netconf-zerotouch]. The on-the-wire pledge to join registrar protocols are different in this protocol from those described in ANIMA, but conceptually operate identically. The vouchers are identical. It is expected that the back-end network operator infrastructure would be able to bootstrap ANIMA-type devices over ethernet, while also being able bootstrap 6tisch devices over 802.15.4 with few changes.

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant STuPiD implementations.

The reader is expected to be familiar with the terms and concepts defined in [I-D.ietf-6tisch-terminology], [RFC7252], [I-D.ietf-core-object-security], and [I-D.ietf-anima-bootstrapping-keyinfra]. The following terms are imported: drop ship, imprint, enrollment, pledge, join proxy, ownership voucher, join registrar/coordinator. The following terms are repeated here for readability, but this document is not authoritative for their definition:

pledge the prospective device, which has the identity provided to at the factory. Neither the device nor the network knows if the device yet knows if this device belongs with this network.

Joined Node the prospective device, after having completing the join process, often just called a Node.

Join Proxy (JP): a stateless relay that provides connectivity between the pledge and the join registrar/coordinator.

Join Registrar/Coordinator (JRC): central entity responsible for authentication and authorization of joining nodes.

Audit Token A signed token from the manufacturer authorized signing authority indicating that the bootstrapping event has been successfully logged. This has been referred to as an "authorization token" indicating that it authorizes bootstrapping to proceed.

Ownership Voucher A signed voucher from the vendor vouching that a specific domain "owns" the new entity as defined in [I-D.ietf-netconf-zero-touch].

MIC manufacturer installed certificate. An [ieee802-1AR] identity.

1.2. Credentials

In the zero-touch scenario, every device expected to be drop shipped would have an [ieee802-1AR] manufacturer installed certificate (MIC). The private key part of the certificate would either be generated in the device, or installed securely (and privately) as part of the manufacturing process. [cullenCiscoPhoneDeploy] provides an example of process which has been active for a good part of a decade.

The MIC would be signed by the manufacturer's CA, the public key component of that would be included in the firmware.

1.2.1. One-Touch Assumptions

This document interacts with the one-touch solution described in [I-D.ietf-6tisch-minimal-security].

1.2.2. Factory provided credentials (if any)

When a manufacturer installed certificate is provided as the IDevID, it SHOULD contain a number of fields. [I-D.ietf-anima-bootstrapping-keyinfra] provides a detailed set of requirements.

A manufacturer unique serial number MUST be provided in the serialNumber SubjectAltName extension, and MAY be repeated in the Common Name. There are no sequential or numeric requirements on the serialNumber, it may be any unique value that the manufacturer wants to use. The serialNumber SHOULD be printed on the packaging and/or on the device in a discrete way so that failures can be physically traced to the relevant device.

1.2.3. Credentials to be introduced

The goal of the bootstrap process is to introduce one or more new locally relevant credentials:

1. a certificate signed by a local certificate authority/registrar. This is the LDevID of [ieee802-1AR].
2. alternatively, a network-wide key to be used to secure L2 traffic.
3. alternatively, a network-wide key to be used to authenticate per-peer keying of L2 traffic using a mechanism such as provided by [ieee802159].

1.3. Network Assumptions

This document is about enrollment of constrained devices [RFC7228] to a constrained network. Constrained networks is such as [ieee802154], and in particular the time-slotted, channel hopping (tsch) mode, feature low bandwidths, and limited opportunities to transmit. A key feature of these networks is that receivers are only listening at certain times.

1.3.1. Security above and below IP

802.15.4 networks have three kinds of layer-2 security:

- o a network key that is shared with all nodes and is used for unicast and multicast. The key may be used for privacy, and it may be used in some cases for authentication only (in the case of enhanced beacons).
- o a series of network keys that are shared (agreed to) between pairs of nodes (the per-peer key)
- o a network key that is shared with all nodes (through a group key management system), and is used for multicast traffic only, while a per-pair key is used for unicast traffic

Setting up the credentials to bootstrap one of these kinds of security, (or directly configuring the key itself for the first case) is required. This is the security below the IP layer.

Security is required above the IP layer: there are three aspects which the credentials in the previous section are to be used.

- o to provide for secure connection with a Path Computation Element [RFC4655], or other LLC (see ({RFC7554}} section 3).
- o to initiate a connection between a Resource Server (RS) and an application layer Authorization Server (AS and CAS from [I-D.ietf-ace-actors]).

1.3.1.1. Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) is the property of a protocol such that complete knowledge of the crypto state (for instance, via a memory dump) at time X does not imply that data from a disjoint time Y can also be recovered. ([PFS]).

PFS is important for two reasons: one is that it offers protection against the compromise of a node. It does this by changing the keys in a non-deterministic way. This second property also makes it much easier to remove a node from the network, as any node which has not participated in the key changing process will find itself no longer connected.

1.3.2. Join network assumptions

The network which the new pledge will connect to will have to have the following properties:

- o a known PANID. The PANID 0xXXXX where XXXX is the assigned RFC# for this document is suggested.
- o a minimal schedule with some Aloha time. This is usually in the same slotframe as the Enhanced Beacon, but a pledge MUST listen for an unencrypted Enhanced Beacon to so that it can synchronize.

1.3.3. Number and cost of round trips

TBD.

1.3.4. Size of packets, number of fragments

1.4. Target end-state for join process

At the end of the zero-touch join process there will be a symmetric key protected channel between the Join Registrar/Coordinator and the pledge, now known as a Joined Node. This channel may be rekeyed via new exchange of asymmetric exponents (ECDH for instance), authenticated using the domain specific credentials created during the join process.

This channel is in the form of an OSCOAP protected connection with [I-D.ietf-core-comi] encoded objects. This document includes definition of a [I-D.ietf-netconf-keystore] compatible objects for encoding of the relevant [I-D.ietf-anima-bootstrapping-keyinfra] objects.

2. Join Protocol

The pledge join protocol state machine is described in [I-D.ietf-6tisch-minimal-security], in section XYZ. The pledge recognizes that it is in zero-touch configuration by the following situation:

- o no PSK has been configured for the network in which it has joined.
- o the pledge has no locally defined certificate (no LDevID), only an IDevID.
- o the network asserts an identity that the pledge does not recognize.

All of these conditions MUST be true. If any of these are not true, then the pledge has either been connected to the wrong network, or it has already been bootstrapped into a different network, and it should wait until it finds that network.

The zero-touch process consists of three stages:

1. the key agreement process
2. the provisional enrollment process
3. the key distribution process

2.1. Key Agreement process

The key agreement process is identical to [I-D.ietf-6tisch-minimal-security]. The process uses EDHOC with certificates.

The pledge will have to trust the JRC provisionally, as described in [I-D.ietf-anima-bootstrapping-keyinfra], section 3.1.2, and in section 4.1.1 of [RFC7030].

The JRC will be able to validate the IDevID of the pledge using the manufacturer's CA.

The pledge may not know if it is in a zero-touch or one-touch situation: the pledge may be able to verify the JRC based upon trust anchors that were installed at manufacturing time. In that case, the pledge runs the simplified one-touch process.

The pledge signals in the EDHOC message_2 if it has accepted the JRC certificate. The JRC will in general, not trust the pledge with the network keys until it has provided the pledge with a voucher. The pledge will notice the absence of the provisioning keys.

XXX - there could be some disconnect here. May need additional signals here.

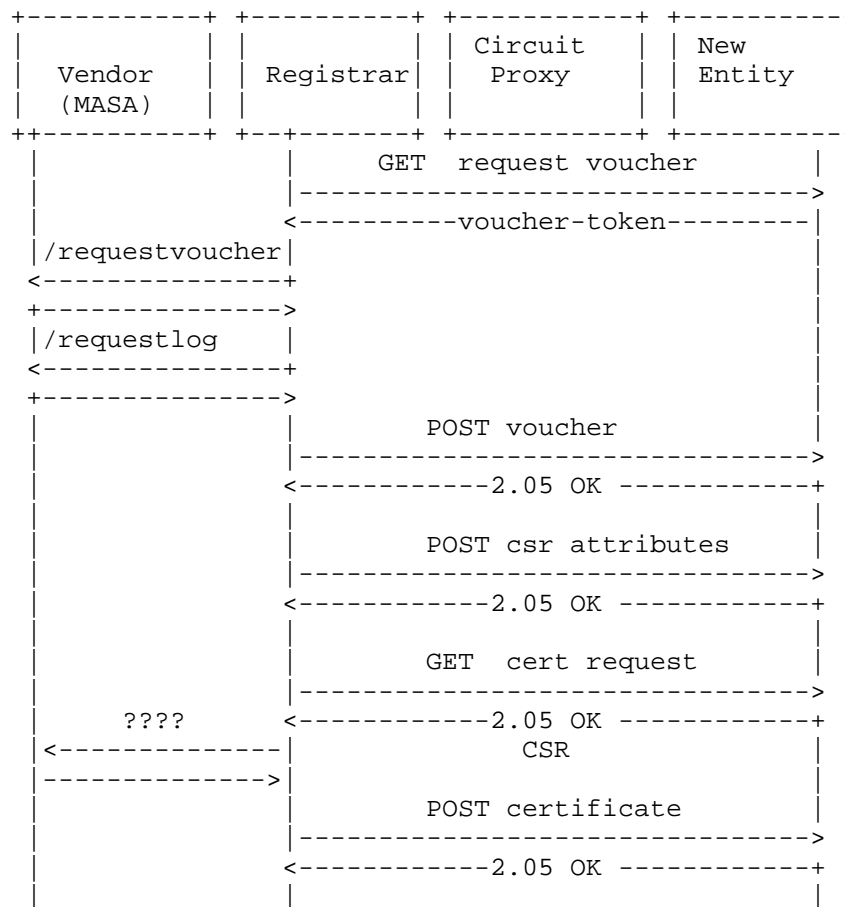
2.2. Provisional Enrollment process

When the pledge determines that it can not verify the certificate of the JRC using built-in trust anchors, then it enters a provisional state. In this state, it keeps the channel created by EDHOC open.

A new EDHOC key derivation is done by the JRC and pledge using a new label, "6tisch-provisional".

The pledge runs as a passive CoMI server, leaving the JRC to drive the enrollment process. The JRC can interrogate the pledge in a variety of fashions as shown below: the process terminates when the JRC provides the pledge with an ownership voucher and the pledge leaves the provisional state.

A typical interaction involves the following requests:



2.3. Key Distribution Process

The key distribution process utilizes the protocol described [I-D.richardson-6tisch-minimal-rekey]. The process starts with the initial key, rather than an actual rekey.

This protocol remains active for subsequent rekey operations.

3. YANG model for BRSKI objects

```
module ietf-6tisch-brski { yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:6tisch-brski"; prefix
  "ietf6brski";
```

```
//import ietf-yang-types { prefix yang; } //import ietf-inet-types {
prefix inet; }

organization "IETF 6tisch Working Group";

contact "WG Web: http://tools.ietf.org/wg/6tisch/ WG List:
6tisch@ietf.org [2] Author: Michael Richardson mcr+ietf@sandelman.ca
[3]";

description "This module defines an interface to set and retrieve
BRSKI objects using CoMI. This interface is used as part of an
enrollment process for constrained nodes and networks.";

revision "2017-03-01" { description "Initial version"; reference "RFC
XXXX: 6tisch zero-touch bootstrap"; }

// top-level container container ietf6brski { leaf requestnonce {
type binary; length XX; // how big can/should it be? mandatory true;
description "Request Nonce."; } leaf voucher { type binary;
description "The voucher as a serialized COSE object"; }

leaf csrattributes {
    type binary;
    description "A list of attributes that MUST be in the CSR";
}

leaf certificaterequest {
    type binary;
    description "A PKIX format Certificate Request";
}

leaf certificate {
    type binary;
    description "The LDevID certificate";
} }
```

3.1. Description of Pledge States in Join Process

TBD

4. Definition of managed objects for zero-touch bootstrap

The following is relevant YANG for use in the bootstrap protocol.
The objects identified are identical in format to the named objects
from [I-D.ietf-anima-bootstrapping-keyinfra].

5. Privacy Considerations

[I-D.ietf-6lo-privacy-considerations] details a number of privacy considerations important in Resource Constrained nodes. There are two networks and three sets of constrained nodes to consider. They are: 1. the production nodes on the production network. 2. the new pledges, which have yet to enroll, and which are on a join network. 3. the production nodes which are also acting as proxy nodes.

5.1. Privacy Considerations for Production network

The details of this are out of scope for this document.

5.2. Privacy Considerations for New Pledges

New Pledges do not yet receive Router Advertisements with PIO options, and so configure link-local addresses only based upon layer-2 addresses using the normal SLAAC mechanisms described in [RFC4191].

These link-local addresses are visible to any on-link eavesdropper (who is synchronized to the same Join Assistant), so regardless of what is chosen they can be seen. This link-layer traffic is encapsulated by the Join Assistant into IPIP packets and carried to the JCE. The traffic SHOULD never leave the operator's network, and no outside traffic should enter, so it should not be possible to do any ICMP scanning as described in [I-D.ietf-6lo-privacy-considerations].

The join process described herein requires that some identifier meaningful to the network operator be communicated to the JCE via the Neighbor Advertisement's ARO option. This need not be a manufacturer created EUI-64 as assigned by IEEE; it could be another value with higher entropy and less interesting vendor/device information. Regardless of what is chosen, it can be used to track where the device attaches.

For most constrained device, network attachment occurs very infrequently, often only once in their lifetime, so tracking opportunities may be rare.

Further, during the enrollment process, a DTLS connection will be created. Unless TLS1.3 is used, the device identity will be visible to passive observers in the 802.11AR IDevID certificate that is sent. Even when TLS1.3 is used, an active attacker could collect the information by simply connecting to the device; it would not have to successfully complete the negotiation either, or even attempt to Man-In-The-Middle the device.

There is, at the same time, significant value in avoiding a link-local DAD process by using an IEEE assigned EUI-64, and there is also significant advantage to the operator being able to see what the vendor of the new device is.

5.2.1. EUI-64 derived address for join time IID

It is therefore suggested that the IID used in the link-local address used during the join process be a vendor assigned EUI-64. After the join process has concluded, the device SHOULD be assigned a unique randomly generated long address, and a unique short address (not based upon the vendor EUI-64) for use at link-layer. At that point, all layer-3 content is encrypted by the layer-2 key.

5.3. Privacy Considerations for Join Assistant

6. Security Considerations

7. IANA Considerations

This document allocates one value from the subregistry "Address Registration Option Status Values": ND_NS_JOIN_DECLINED Join Assistant, JOIN_DECLINED (TBD-AA)

8. Protocol Definition

9. Acknowledgements

Kristofer Pister helped with many non-IETF references.

10. References

10.1. Normative References

- [cullenCiscoPhoneDeploy]
Jennings, C., "Transitive Trust Enrollment for Constrained Devices", 2012, <<http://www.lix.polytechnique.fr/hipercom/SmartObjectSecurity/papers/CullenJennings.pdf>>.
- [I-D.ietf-6lo-privacy-considerations]
Thaler, D., "Privacy Considerations for IPv6 Adaptation Layer Mechanisms", draft-ietf-6lo-privacy-considerations-04 (work in progress), October 2016.
- [I-D.ietf-6tisch-minimal]
Vilajosana, X., Pister, K., and T. Watteyne, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-21 (work in progress), February 2017.

- [I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., and K. Pister, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-01 (work in progress), February 2017.
- [I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-08 (work in progress), December 2016.
- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-04 (work in progress), October 2016.
- [I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-09 (work in progress), December 2016.
- [I-D.ietf-core-comi]
Stok, P., Bierman, A., Veillette, M., and A. Pelov, "CoAP Management Interface", draft-ietf-core-comi-00 (work in progress), January 2017.
- [I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", draft-ietf-core-object-security-01 (work in progress), December 2016.
- [I-D.ietf-netconf-keystore]
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.
- [I-D.ietf-netconf-zerotouch]
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-12 (work in progress), January 2017.
- [I-D.richardson-6tisch-join-enhanced-beacon]
Richardson, M., "802.15.4 Informational Element encapsulation of 6tisch Join Information", draft-richardson-6tisch-join-enhanced-beacon-00 (work in progress), February 2017.

- [I-D.richardson-6tisch-minimal-rekey]
Richardson, M., "Minimal Security rekeying mechanism for 6TiSCH", draft-richardson-6tisch-minimal-rekey-00 (work in progress), February 2017.
- [I-D.richardson-anima-6join-discovery]
Richardson, M., "GRASP discovery of Registrar by Join Assistant", draft-richardson-anima-6join-discovery-00 (work in progress), October 2016.
- [iec62591]
IEC, ., "62591:2016 Industrial networks - Wireless communication network and communication profiles - WirelessHART", 2016, <<https://webstore.iec.ch/publication/24433>>.
- [ieee802-1AR]
IEEE Standard, ., "IEEE 802.1AR Secure Device Identifier", 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [ieee802154]
IEEE Standard, ., "802.15.4-2015 - IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015, <<http://standards.ieee.org/findstds/standard/802.15.4-2015.html>>.
- [ieee802159]
IEEE Standard, ., "802.15.9-2016 - IEEE Approved Draft Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams", 2016, <<http://standards.ieee.org/findstds/standard/802.15.9-2016.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<http://www.rfc-editor.org/info/rfc6775>>.

- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<http://www.rfc-editor.org/info/rfc7030>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<http://www.rfc-editor.org/info/rfc7217>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<http://www.rfc-editor.org/info/rfc7228>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<http://www.rfc-editor.org/info/rfc7252>>.

10.2. Informative References

- [duckling] Stajano, F. and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", 1999, <<https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>>.
- [I-D.ietf-ace-actors] Gerdes, S., Seitz, L., Selander, G., and C. Bormann, "An architecture for authorization in constrained environments", draft-ietf-ace-actors-04 (work in progress), September 2016.
- [I-D.ietf-roll-useofrplinfo] Robles, I., Richardson, M., and P. Thubert, "When to use RFC 6553, 6554 and IPv6-in-IPv6", draft-ietf-roll-useofrplinfo-10 (work in progress), December 2016.
- [ISA100] "The Technology Behind the ISA100.11a Standard", June 2010, <http://www.isa100wci.org/Documents/PDF/The-Technology-Behind-ISA100-11a-v-3_pptx>.
- [PFS] Wikipedia, ., "Forward Secrecy", August 2016, <https://en.wikipedia.org/w/index.php?title=Forward_secrecy&oldid=731318899>.

- [pledge] Dictionary.com, ., "Dictionary.com Unabridged", 2015, <<http://dictionary.reference.com/browse/pledge>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<http://www.rfc-editor.org/info/rfc4655>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<http://www.rfc-editor.org/info/rfc7554>>.
- [RFC7731] Hui, J. and R. Kelsey, "Multicast Protocol for Low-Power and Lossy Networks (MPL)", RFC 7731, DOI 10.17487/RFC7731, February 2016, <<http://www.rfc-editor.org/info/rfc7731>>.

10.3. URIs

[2] <mailto:6tisch@ietf.org>

[3] <mailto:mcr+ietf@sandelman.ca>

Appendix A. appendix

insert appendix here

Author's Address

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2017

M. Vucinic, Ed.
Inria
J. Simon
Linear Technology
K. Pister
University of California Berkeley
M. Richardson
Sandelman Software Works
June 15, 2017

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-03

Abstract

This document describes the minimal mechanisms required to support secure enrollment of a pledge, a device being added to an IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) network. It assumes that the pledge has been provisioned with a credential that is relevant to the deployment - the "one-touch" scenario. The goal of this configuration is to set link-layer keys, and to establish a secure end-to-end session between each pledge and the join registrar who may use that to further configure the pledge. Additional security behaviors and mechanisms may be added on top of this minimal framework.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 17, 2017.

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. Terminology	3
3. One-Touch Assumptions	4
4. Join Overview	4
4.1. Step 1 - Enhanced Beacon	5
4.2. Step 2 - Neighbor Discovery	6
4.3. Step 3 - Security Handshake	6
4.4. Step 4 - Simple Join Protocol - Join Request	8
4.5. Step 5 - Simple Join Protocol - Join Response	8
5. Architectural Overview and Communication through Join Proxy	9
5.1. Stateless-Proxy CoAP Option	9
6. Security Handshake	10
7. Simple Join Protocol Specification	11
7.1. OSCOAP Security Context Instantiation	12
7.2. Specification of Join Request	13
7.3. Specification of Join Response	13
8. Mandatory to Implement Algorithms and Certificate Format	15
9. Link-layer Requirements	15
10. Rekeying and Rejoin	16
11. Key Derivations	16
12. Security Considerations	16
13. Privacy Considerations	17
14. IANA Considerations	18
14.1. CoAP Option Numbers Registry	18
15. Acknowledgments	18
16. References	18
16.1. Normative References	19
16.2. Informative References	19
Appendix A. Example	21
Authors' Addresses	23

1. Introduction

This document describes the minimal feature set for a new device, termed pledge, to securely join a 6TiSCH network. As a successful outcome of this process, the pledge is able to securely communicate with its neighbors, participate in the routing structure of the network or establish a secure session with an Internet host.

When a pledge seeks admission to a 6TiSCH [RFC7554] network, it first needs to synchronize to the network. The pledge then configures its link-local IPv6 address and authenticates itself, and also validates that it is joining the right network. At this point it can expect to interact with the network to configure its link-layer keying material. Only then may the node establish an end-to-end secure session with an Internet host using OSCOAP [I-D.ietf-core-object-security] or DTLS [RFC6347]. Once the application requirements are known, the node interacts with its peers to request additional resources as needed, or to be reconfigured as the network changes [I-D.ietf-6tisch-6top-protocol].

This document presumes a network as described by [RFC7554], [I-D.ietf-6tisch-6top-protocol], and [I-D.ietf-6tisch-terminology]. It assumes the pledge pre-configured with either a:

- o pre-shared key (PSK),
- o raw public key (RPK),
- o or a locally-valid certificate and a trust anchor.

As the outcome of the join process, the pledge expects one or more link-layer key(s) and optionally a temporary link-layer identifier.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words may also appear in this document in lowercase, absent their normative meanings.

The reader is expected to be familiar with the terms and concepts defined in [I-D.ietf-6tisch-terminology], [RFC7252], [I-D.ietf-core-object-security], and [I-D.ietf-anima-bootstrapping-keyinfra]. The following terms are imported: pledge, join proxy, join registrar/coordinator, drop ship, imprint, enrollment, ownership voucher.

Pledge: the prospective device, which has the identity provided to at the factory.

Joined Node: the prospective device, after having completed the join process, often just called a Node.

Join Proxy (JP): a stateless relay that provides connectivity between the pledge and the Join Registrar/Coordinator.

Join Registrar/Coordinator (JRC): central entity responsible for authentication and authorization of joining nodes.

3. One-Touch Assumptions

This document assumes the one-touch scenario, where devices are provided with some mechanism by which a secure association may be made in a controlled environment. There are many ways in which this might be done, and detailing any of them is out of scope for this document. But, some notion of how this might be done is important so that the underlying assumptions can be reasoned about.

Some examples of how to do this could include:

- o JTAG interface
- o serial (craft) console interface
- o pushes of physical buttons simultaneous to network attachment
- o unsecured devices operated in a Faraday cage

There are likely many other ways as well. What is assumed is that there can be a secure, private conversation between the Join Registrar/Coordinator, and the pledge, and that the two devices can exchange some trusted bytes of information.

4. Join Overview

This section describes the steps taken by a pledge in a 6TiSCH network. When a previously unknown device seeks admission to a 6TiSCH [RFC7554] network, the following exchange occurs:

1. The pledge listens for an Enhanced Beacon (EB) frame [IEEE8021542015]. This frame provides network synchronization information, and tells the device when it can send a frame to the node sending the beacons, which plays the role of Join Proxy (JP) for the pledge, and when it can expect to receive a frame.

2. The pledge configures its link-local IPv6 address and advertizes it to Join Proxy (JP).
3. The pledge sends packets to JP in order to securely identify itself to the network. These packets are directed to the Join Registrar/Coordinator (JRC), which may be co-located on the JP or another device.
4. The pledge receives one or more packets from JRC (via the JP) that sets up one or more link-layer keys used to authenticate subsequent transmissions to peers.

From the pledge's perspective, minimal joining is a local phenomenon - the pledge only interacts with the JP, and it need not know how far it is from the 6LBR, or how to route to the JRC. Only after establishing one or more link-layer keys does it need to know about the particulars of a 6TiSCH network.

The handshake is shown as a transaction diagram in Figure 1:

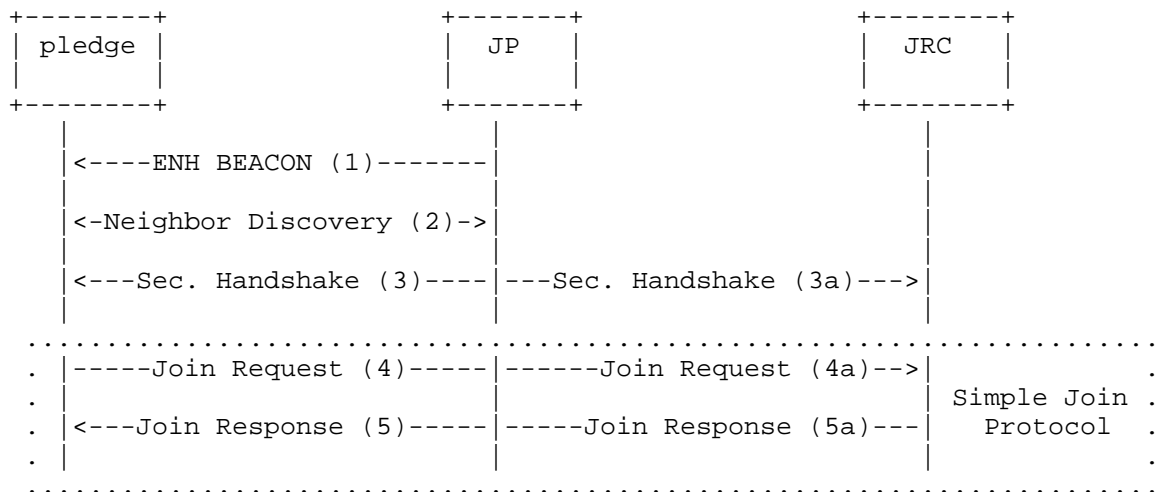


Figure 1: Overview of the join process.

The details of each step are described in the following sections.

4.1. Step 1 - Enhanced Beacon

Due to the channel hopping nature of 6TiSCH, transmissions take place on physical channels in a circular fashion. For that reason, Enhanced Beacons (EBs) are expected to be found by listening on a single channel. However, because some channels may be blacklisted, a

new pledge must listen for Enhanced Beacons for a certain period on each of the 16 possible channels. This search process entails having the pledge keep the receiver portion of its radio active for the entire period of time.

Once the pledge hears an EB from a JP, it synchronizes itself to the joining schedule using the cells contained in the EB. The selection of which beacon to start with is outside the scope of this document. Implementers SHOULD make use of information such as: whether the L2 address of the EB has been tried before, any Network Identifier [I-D.richardson-6tisch-join-enhanced-beacon] seen, and the strength of the signal. The pledge can be configured with the Network Identifier to seek when it is configured with the PSK.

Once a candidate network has been selected, the pledge can transition into a low-power duty cycle, waking up only when the provided schedule indicates shared slots which the pledge may use for the join process.

At this point the pledge may proceed to step 2, or continue to listen for additional EBs.

A pledge which receives only Enhanced Beacons containing Network ID extensions [I-D.richardson-6tisch-join-enhanced-beacon] with the initiate bit cleared, SHOULD NOT proceed with this protocol on that network. The pledge SHOULD consider that it is in a network which manages join traffic, it SHOULD switch to [I-D.ietf-6tisch-dtsecurity-secure-join].

4.2. Step 2 - Neighbor Discovery

At this point, the pledge forms its link-local IPv6 address based on EUI64 and may register it at JP, in order to bootstrap the IPv6 neighbor tables. The Neighbor Discovery exchange shown in Figure 1 refers to a single round trip Neighbor Solicitation / Neighbor Advertisement exchange between the pledge and the JP. The pledge may further follow the Neighbor Discovery (ND) process described in Section 5 of [RFC6775].

4.3. Step 3 - Security Handshake

The security handshake between pledge and JRC uses Ephemeral Diffie-Hellman over COSE (EDHOC) [I-D.selander-ace-cose-ecdhe] to establish the shared session secret used to encrypt the Simple Join Protocol.

The security handshake step is OPTIONAL in case PSKs are used, while it is REQUIRED for RPKs and certificates.

When using certificates, the process continues as described in [I-D.selander-ace-cose-ecdhe], but MAY result in no network key being returned. In that case, the pledge enters a provisional situation where it provides access to an enrollment mechanism described in [I-D.ietf-6tisch-dtsecurity-secure-join].

If using a locally relevant certificate, the pledge will be able to validate the certificate of the JRC via a local trust anchor. In that case, the JRC will return network keys as in the PSK case. This would typically be the case for a device which has slept so long that it no longer has valid network keys and must go through a partial join process again.

In case the handshake step is omitted, the shared secret used for protection of the Simple Join Protocol in the next step is the PSK.

A consequence is that if the long-term PSK is compromised, keying material transferred as part of the join response is compromised as well. Physical compromise of the pledge, however, would also imply the compromise of the same keying material, as it is likely to be found in node's memory.

4.3.1. Pre-Shared Symmetric Key

The Diffie-Hellman key exchange and the use of EDHOC is optional, when using a pre-shared symmetric key. This cuts down on traffic between JRC and pledge, but requires pre-configuration of the shared key on both devices.

It is REQUIRED to use unique PSKs for each pledge. If there are multiple JRCs in the network (such as for redundancy), they would have to share a database of PSKs.

4.3.2. Asymmetric Keys

The Security Handshake step is required, when using asymmetric keys. Before conducting the Diffie-Hellman key exchange using EDHOC [I-D.selander-ace-cose-ecdhe] the pledge and JRC need to receive and validate each other's public key certificate. As detailed above, this can only be done for locally relevant (LDevID) certificates. IDevID certificates require entering a provisional state as described in [I-D.ietf-6tisch-dtsecurity-secure-join].

When RPKs are pre-configured at pledge and JRC, they can directly proceed to the handshake.

4.4. Step 4 - Simple Join Protocol - Join Request

The Join Request that makes part of the Simple Join Protocol is sent from the pledge to the JP using the shared slot as described in the EB, and forwarded to the JRC. Which slot the JP uses to transmit to the JRC is out of scope: some networks may wish to dedicate specific slots for this join traffic.

The join request is authenticated/encrypted end-to-end using an algorithm from [I-D.ietf-cose-msg] and a key derived from the shared secret from step 3. Algorithm negotiation is described in detail in [I-D.selander-ace-cose-ecdhe], and mandatory to implement algorithms are specified in Section 8.

The nonce is derived from the shared secret, the pledge's EUI64 and a monotonically increasing counter initialized to 0 when first starting.

4.5. Step 5 - Simple Join Protocol - Join Response

The Join Response that makes part of the Simple Join Protocol is sent from the JRC to the pledge through JP that serves as a stateless relay. Packet containing the Join Response travels on the path from JRC to JP using pre-established routes in the network. The JP delivers it to the pledge using the slot information from the EB. JP operates as the application-layer proxy and does not keep any state to relay the message. It uses information sent in the clear within the join response to decide where to forward to.

The join response is authenticated/encrypted end-to-end using an algorithm from [I-D.ietf-cose-msg] and a key derived from the shared secret from step 3.

The nonce is derived from the shared secret, pledge's EUI64 and a monotonically increasing counter matching that of the join request.

The join response contains one or more link-layer key(s) that the pledge will use for subsequent communication. Each key that is provided by the JRC is associated with an 802.15.4 key identifier. In other link-layer technologies, a different identifier may be substituted. Join Response optionally also contains an IEEE 802.15.4 short address [IEEE8021542015] assigned to pledge by JRC, and the IPv6 address of the JRC.

5. Architectural Overview and Communication through Join Proxy

The protocol in Figure 1 is implemented over Constrained Application Protocol (CoAP) [RFC7252]. The Pledge plays the role of a CoAP client, JRC the role of a CoAP server, while JP implements CoAP forward proxy functionality [RFC7252]. Since JP is also likely a constrained device, it does not need to implement a cache but rather process forwarding-related CoAP options and make requests on behalf of pledge that is not yet part of the network.

The pledge communicates with a Join Proxy (JP) over link-local IPv6 addresses. The pledge designates a JP as a proxy by including in the CoAP requests to the JP the Proxy-Scheme option with value "coap" (CoAP-to-CoAP proxy). The pledge MUST include the Uri-Host option with its value set to the well-known JRC's alias - "6tisch.arpa". The pledge learns the actual IPv6 address of JRC from the join response and it uses it once joined in order to operate as JP. The initial bootstrap of the 6LBR would require explicit provisioning of the JRC address.

5.1. Stateless-Proxy CoAP Option

The CoAP proxy by default keeps per-client state information in order to forward the response towards the originator of the request (client). This state information comprises CoAP token, but the implementations also need to keep track of the IPv6 address of the host, as well as the corresponding UDP source port number. In the setting where the proxy is a constrained device and there are potentially many clients, as in the case of JP, this makes it prone to Denial of Service (DoS) attacks, due to the limited memory.

The Stateless-Proxy CoAP option (c.f. Figure 2) allows the proxy to insert within the request the state information necessary for relaying the response back to the client. Note that the proxy still needs to keep some state, such as for performing congestion control or request retransmission, but what is aimed with Stateless-Proxy option is to free the proxy from keeping per-client state.

Stateless-Proxy option is critical, Safe-to-Forward, not part of the cache key, not repeatable and opaque. When processed by OSCOAP, Stateless-Proxy option is neither encrypted nor integrity protected.

No.	C	U	N	R	Name	Format	Length
TBD	x		x		Stateless-Proxy	opaque	1-255

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Figure 2: Stateless-Proxy CoAP Option

Upon reception of a Stateless-Proxy option, the CoAP server MUST echo it in the response. The value of the Stateless-Proxy option is internal proxy state that is opaque to the server. Example state information includes IPv6 address of the client, its UDP source port, and the CoAP token. For security reasons, the state information MUST be authenticated, MUST include a freshness indicator (e.g. a sequence number or timestamp) and MAY be encrypted. The proxy may use an appropriate COSE structure [I-D.ietf-cose-msg] to wrap the state information as the value of the Stateless-Proxy option. The key used for encryption/authentication of the state information may be known only to the proxy.

Once the proxy has received the CoAP response with Stateless-Proxy option present, it decrypts/authenticates it, checks the freshness indicator and constructs the response for the client, based on the information present in the option value.

Note that a CoAP proxy using the Stateless-Proxy option is not able to return 5.04 Gateway Timeout error in case the request to the server times out. Likewise, if the response to the proxy's request does not contain the Stateless-Proxy option, for example when the option is not supported by the server, the proxy is not able to return the response to the client.

6. Security Handshake

In order to derive a shared session key, pledge and JRC run the EDHOC protocol [I-D.selander-ace-cose-ecdhe]. During this process, pledge and JRC mutually authenticate each other and verify authorization information before proceeding with the Simple Join Protocol. In case certificates are used for authentication, this document assumes that a special certificate with role attribute set has been provisioned to the JRC. This certificate is verified by pledge in order to authorize JRC to continue with the join process. How such a certificate is issued to the JRC is out of scope of this document.

Figure 3 details the exchanges between the pledge and JRC that take place during the execution of the security handshake. Format of EDHOC messages is specified in [I-D.selander-ace-cose-ecdhe]. The

handshake is initiated by the pledge. JRC may either respond with an empty CoAP acknowledgment, signaling to the pledge that it needs to wait, or directly with the second message of EDHOC handshake. How JRC decides whether it will immediately proceed with the handshake is out of scope of this document.

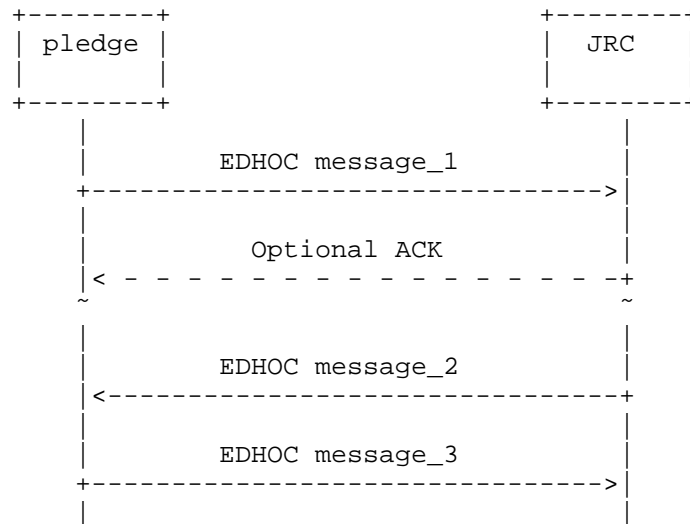


Figure 3: Transaction diagram of the security handshake.

7. Simple Join Protocol Specification

Simple Join Protocol is a single round trip protocol (c.f. Figure 4) that facilitates secure enrollment of a pledge, based on a shared symmetric secret. In case the pledge was provisioned by an asymmetric key (certificate or RPK), Simple Join Protocol is preceded by a security handshake, described in Section 6. When the pledge is provisioned with a PSK, Simple Join Protocol may be run directly.

Pledge and JRC MUST protect their exchange end-to-end (i.e. through the proxy) using Object Security of CoAP (OSCOAP) [I-D.ietf-core-object-security].

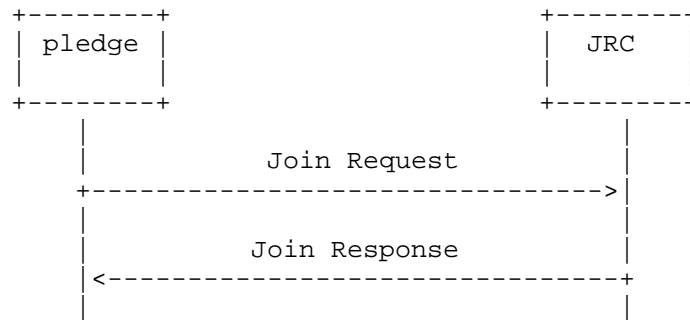


Figure 4: Transaction diagram of the Simple Join Protocol.

7.1. OSCOAP Security Context Instantiation

The OSCOAP security context MUST be derived at pledge and JRC as per Section 3.2 of [I-D.ietf-core-object-security] using HKDF SHA-256 [RFC5869] as the key derivation function.

- o Master Secret MUST be the secret generated by the run of EDHOC as per Appendix B of [I-D.selander-ace-cose-ecdhe], or the PSK in case EDHOC step was omitted.
- o Sender ID of the pledge MUST be set to the concatenation of its EUI-64 and byte string 0x00.
- o Recipient ID (ID of JRC) MUST be set to the concatenation of pledge's EUI-64 and byte string 0x01. The construct uses pledge's EUI-64 to avoid nonce reuse in the response in the case same PSK is shared by a group of pledges.
- o Algorithm MUST be set to the value from [I-D.ietf-cose-msg] agreed by the run of EDHOC, or out-of-band in case of PSKs.

The derivation in [I-D.ietf-core-object-security] results in traffic keys and static IVs for each side of the conversation. Nonces are constructed by XOR'ing the static IV with current sequence number. The context derivation process occurs exactly once.

Implementations MUST ensure that multiple CoAP requests to different JRCs result in the use of the same OSCOAP context so that sequence numbers are properly incremented for each request. This may happen in a scenario where there are multiple 6TiSCH networks present and the pledge tries to join one network at a time.

7.2. Specification of Join Request

Message Join Request SHALL be mapped to a CoAP request:

- o The request method is GET.
- o The Proxy-Scheme option is set to "coap".
- o The Uri-Host option is set to "6tisch.arpa".
- o The Uri-Path option is set to "j".
- o The object security option SHALL be set according to [I-D.ietf-core-object-security] and OSCOAP parameters set as described above.

7.3. Specification of Join Response

If OSCOAP processing is a success and the pledge is authorized to join the network, message Join Response SHALL be mapped to a CoAP response:

- o The response Code is 2.05 (Content).
- o Content-Format option is set to application/cbor.
- o The payload is a CBOR [RFC7049] array containing, in order:
 - * COSE Key Set, specified in [I-D.ietf-cose-msg], containing one or more link-layer keys. The mapping of individual keys to 802.15.4-specific parameters is described in Section 7.3.1.
 - * Optional. Link layer short address that is assigned to the pledge. The format of the short address follows Section 7.3.2.
 - * Optional. IPv6 address of the JRC transported as a byte string. If the address of the JRC is not present in the response, JRC is co-located with 6LBR.

```
payload = [  
    COSE_KeySet,  
    ? short_address,  
    ? JRC_address : bstr,  
]
```

7.3.1. Link-layer Keys Transported in COSE Key Set

Each key in the COSE Key Set [I-D.ietf-cose-msg] SHALL be a symmetric key. If "kid" parameter of the COSE Key structure is present, the corresponding keys SHALL belong to an IEEE 802.15.4 KeyIdMode 0x01 class. In that case, parameter "kid" of COSE Key structure SHALL be used to carry IEEE 802.15.4 KeyIndex value. If the "kid" parameter is not present in the transported key, the application SHALL consider the key to be an IEEE 802.15.4 KeyIdMode 0x00 (implicit) key. This document does not support IEEE 802.15.4 KeyIdMode 0x02 and 0x03 class keys.

7.3.2. Short Address

Optional "short_address" structure transported as part of the join response payload represents IEEE 802.15.4 short address assigned to the pledge. It is encoded as CBOR array object, containing in order:

- o Byte string, containing the 16-bit address.
- o Optional lease time parameter, "lease_asn". The value of the "lease_asn" parameter is the 5-byte Absolute Slot Number (ASN) corresponding to its expiration, carried as a byte string in network byte order.

```
short_address = [  
    address : bstr,  
    ? lease_asn : bstr,  
]
```

It is up to the joined node to request a new short address before the expiry of its previous address. The mechanism by which the node requests renewal is the same as during join procedure, as described in Section 10. The assigned short address is used for configuring both Layer 2 short address and Layer 3 addresses.

7.3.3. Error Handling

In the case JRC determines that pledge is not supposed to join the network (e.g. by failing to find an appropriate security context), it should respond with a 4.01 Unauthorized error. Upon reception of a 4.01 Unauthorized, the pledge SHALL attempt to join the next advertised 6TiSCH network. If all join attempts have failed at pledge, the pledge SHOULD signal to the user by an out-of-band mechanism the presence of an error condition.

In the case that the JRC determines that the pledge is not (yet) authorized to join the network, but a further zero-touch process

might permit it, the JRC responds with a 2.05 (Content) code, but the payload contains the single CBOR string "prov" (for "provisional"). No link-layer keys or short address is returned.

This response is typically only expected when in asymmetric certificate mode using 802.1AR IDevID certificates. But for reasons of provisioning or device reuse, this could occur even when a one-touch PSK authentication process was expected.

8. Mandatory to Implement Algorithms and Certificate Format

The mandatory to implement symmetric-key algorithm for use with OSCOAP is AES-CCM-16-64-128 from [I-D.ietf-cose-msg]. This is the algorithm used in 802.15.4, and is present in hardware on many platforms. With this choice, CoAP messages are therefore protected with an 8-byte CCM authentication tag and the algorithm uses 13-byte long nonces.

The mandatory to implement hash algorithm is SHA-256 [RFC4231].

Certificates or pre-configured RPKs may be used to exchange public keys between the pledge and JRC. The mandatory to implement Elliptic Curve is P-256, also known as secp256r1. The mandatory to implement signature algorithm is ECDSA with SHA-256.

The certificate itself may be a compact representation of an X.509 certificate, or a full X.509 certificate. Compact representation of X.509 certificates is out of scope of this specification. The certificate is signed by a root CA whose certificate is installed on all nodes participating in a particular 6TiSCH network, allowing each node to validate the certificate of the JRC or pledge as appropriate.

9. Link-layer Requirements

In an operational 6TiSCH network, all frames MUST use link-layer frame security. The frame security options MUST include frame authentication, and MAY include frame encryption.

Link-layer frames are protected with a 16-byte key, and a 13-byte nonce constructed from current Absolute Slot Number (ASN) and the source (the JP for EBs) address, as shown in Figure 5:

```
+-----+
| Address (8B or 00-padded 2B) | ASN (5B) |
+-----+
```

Figure 5: Link-layer CCM* nonce construction

The pledge does not initially do any authentication of the EB frames, as it does not know the K1 key. When sending frames, the pledge sends unencrypted and unauthenticated frames. JP accepts these frames (exempt mode in 802.15.4) for the duration of the join process. How JP learns whether the join process is ongoing is out of scope of this specification.

As the EB itself cannot be authenticated by pledge, an attacker may craft a frame that appears to be a valid EB, since the pledge can neither know the ASN a priori nor verify the address of the JP. This permits a Denial of Service (DoS) attack at the pledge. Beacon authentication keys are discussed in [I-D.ietf-6tisch-minimal].

10. Rekeying and Rejoin

This protocol handles initial keying of the pledge. For reasons such as rejoining after a long sleep, or expiry of the short address, the joined node MAY send a new Join Request over the previously established secure end-to-end session with JRC. JRC responds with up-to-date keys and a short address. The node may also use the Simple Join Protocol exchange for node-initiated rekeying. How node learns that it should be rekeyed is out of scope. Additional work, such as in [I-D.richardson-6tisch-minimal-rekey] can be used.

11. Key Derivations

When EDHOC is used to derive keys, the cost of the asymmetric operation can be amortized over any additional connections that may be required between the node (during or after joining) and the JRC.

Each application SHOULD use a unique session key. EDHOC was designed with this in mind. In order to accomplish this, the EDHOC key derivation algorithm can be run with a different label. Other users of this key MUST define the label.

12. Security Considerations

In case PSKs are used, this document mandates that the pledge and JRC are pre-configured with unique keys. The uniqueness of generated nonces is guaranteed under the assumption of unique EUI64 identifiers for each pledge. Note that the address of the JRC does not take part in nonce construction. Therefore, even should an error occur, and a PSK shared by a group of nodes, the nonces constructed as part of the different responses are unique. The PSK is still important for authentication of the pledge and authentication of the JRC to the pledge. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible. The well known problem with Bluetooth headsets with a "0000" pin applies here. The design differentiates

between nonces constructed for requests and nonces constructed for responses by different sender identifiers (0x00 for pledge and 0x01 for JRC).

Being a stateless relay, JP blindly forwards the join traffic into the network. While the exchange between pledge and JP takes place over a shared cell, join traffic is forwarded using dedicated cells on the JP to JRC path. In case of distributed scheduling, the join traffic may therefore cause intermediate nodes to request additional bandwidth. (EDNOTE: this is a problem that needs to be solved) Because the relay operation of JP is implemented at the application layer, JP is the only hop on the JP-6LBR path that can distinguish join traffic from regular IP traffic in the network. It is therefore recommended to implement stateless rate limiting at JP: a simple bandwidth (in bytes or packets/second) cap would be appropriate.

The shared nature of the "minimal" cell used for join traffic makes the network prone to DoS attacks by congesting the JP with bogus radio traffic. As such an attacker is limited by emitted radio power, redundancy in the number of deployed JPs alleviates the issue and also gives the pledge a possibility to use the best available link for join. How a network node decides to become a JP is out of scope of this specification.

At the time of the join, the pledge has no means of verifying the content in the EB and has to accept it at "face value". In case the pledge tries to join an attacker's network, the join response message in such cases will either fail the security check or time out. The pledge may implement a blacklist in order to filter out undesired beacons and try to join the next seemingly valid network. The blacklist alleviates the issue but is effectively limited by the node's available memory. Such bogus beacons will prolong the join time of the pledge and so the time spent in "minimal" [I-D.ietf-6tisch-minimal] duty cycle mode.

13. Privacy Considerations

This specification relies on the uniqueness of EUI64 that is transferred in clear as part of the security context identifier. (EDNOTE: should we say IID here?) Privacy implications of using such long-term identifier are discussed in [RFC7721] and comprise correlation of activities over time, location tracking, address scanning and device-specific vulnerability exploitation. Since the join protocol is executed rarely compared to the network lifetime, long-term threats that arise from using EUI64 are minimal. In addition, the join response message contains an optional short address which can be assigned by JRC to the pledge. The short address is independent of the long-term identifier EUI64 and is

encrypted in the response. For that reason, it is not possible to correlate the short address with the EUI64 used during the join. Use of short addresses once the join protocol completes mitigates the aforementioned privacy risks. In addition, EDHOC may be used for identity protection during the join protocol by generating a random context identifier in place of the EUI64 [I-D.selander-ace-cose-ecdhe].

14. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "[[this document]]" with the RFC number of this specification.

This document allocates a well-known name under the .arpa name space according to the rules given in: [RFC3172]. The name "6tisch.arpa" is requested. No subdomains are expected. No A, AAAA or PTR record is requested.

14.1. CoAP Option Numbers Registry

The Stateless-Proxy option is added to the CoAP Option Numbers registry:

Number	Name	Reference
TBD	Stateless-Proxy	[[this document]]

15. Acknowledgments

The work on this document has been partially supported by the European Union's H2020 Programme for research, technological development and demonstration under grant agreement No 644852, project ARMOUR.

The authors are grateful to Thomas Watteyne and Goeran Selander for reviewing the draft and to Klaus Hartke for providing input on the Stateless-Proxy CoAP option. The authors would also like to thank Francesca Palombini and Ludwig Seitz for participating in the discussions that have helped shape the document.

16. References

16.1. Normative References

- [I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz,
"Object Security of CoAP (OSCOAP)", draft-ietf-core-
object-security-03 (work in progress), May 2017.
- [I-D.ietf-cose-msg]
Schaad, J., "CBOR Object Signing and Encryption (COSE)",
draft-ietf-cose-msg-24 (work in progress), November 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3172] Huston, G., Ed., "Management Guidelines & Operational
Requirements for the Address and Routing Parameter Area
Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172,
September 2001, <<http://www.rfc-editor.org/info/rfc3172>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object
Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
October 2013, <<http://www.rfc-editor.org/info/rfc7049>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
Application Protocol (CoAP)", RFC 7252,
DOI 10.17487/RFC7252, June 2014,
<<http://www.rfc-editor.org/info/rfc7252>>.

16.2. Informative References

- [I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol
(6P)", draft-ietf-6tisch-6top-protocol-05 (work in
progress), May 2017.
- [I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-
6tisch-dtsecurity-secure-join-01 (work in progress),
February 2017.
- [I-D.ietf-6tisch-minimal]
Vilajosana, X., Pister, K., and T. Watteyne, "Minimal
6TiSCH Configuration", draft-ietf-6tisch-minimal-21 (work
in progress), February 2017.

- [I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang,
"Terminology in IPv6 over the TSCH mode of IEEE
802.15.4e", draft-ietf-6tisch-terminology-08 (work in
progress), December 2016.
- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
S., and K. Watsen, "Bootstrapping Remote Secure Key
Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
keyinfra-06 (work in progress), May 2017.
- [I-D.richardson-6tisch-join-enhanced-beacon]
Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational
Element encapsulation of 6tisch Join Information", draft-
richardson-6tisch-join-enhanced-beacon-01 (work in
progress), March 2017.
- [I-D.richardson-6tisch-minimal-rekey]
Richardson, M., "Minimal Security rekeying mechanism for
6TiSCH", draft-richardson-6tisch-minimal-rekey-01 (work in
progress), February 2017.
- [I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral
Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-
cose-ecdhe-06 (work in progress), April 2017.
- [IEEE8021542015]
IEEE standard for Information Technology, ., "IEEE Std
802.15.4-2015 Standard for Low-Rate Wireless Personal Area
Networks (WPANs)", 2015.
- [RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-
224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512",
RFC 4231, DOI 10.17487/RFC4231, December 2005,
<<http://www.rfc-editor.org/info/rfc4231>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand
Key Derivation Function (HKDF)", RFC 5869,
DOI 10.17487/RFC5869, May 2010,
<<http://www.rfc-editor.org/info/rfc5869>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.

- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<http://www.rfc-editor.org/info/rfc6775>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<http://www.rfc-editor.org/info/rfc7554>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<http://www.rfc-editor.org/info/rfc7721>>.

Appendix A. Example

Figure 6 illustrates a join protocol exchange in case PSKs are used. Pledge instantiates the OSCOAP context and derives the traffic keys and nonces from the PSK. It uses the instantiated context to protect the CoAP request addressed with Proxy-Scheme option and well-known host name of JRC in the Uri-Host option. Triggered by the presence of Proxy-Scheme option, JP forwards the request to the JRC and adds the Stateless-Proxy option with value set to the internally needed state, authentication tag, and a freshness indicator. JP learned the IPv6 address of JRC when it acted as a pledge and joined the network. Once JRC receives the request, it looks up the correct context based on the Sender ID (sid) parameter. It reconstructs OSCOAP's external Additional Authenticated Data (AAD) needed for verification based on:

- o Version field of the received CoAP header.
- o Code field of the received CoAP header.
- o Algorithm being the AES-CCM-16-64-128 from [I-D.ietf-cose-msgl].
- o Request ID being set to pledge's EUI-64 concatenated with 0x00.
- o Request Sequence number set to the value of "Partial IV" of the received COSE object.

Replay protection is ensured by OSCOAP and the tracking of sequence numbers at each side. In the example below, the response contains sequence number 7 meaning that there have already been some attempts to join under a given context, not coming from the pledge. Once JP receives the response, it authenticates the Stateless-Proxy option

before deciding where to forward. JP sets its internal state to that found in the Stateless-Proxy option. Note that JP does not possess the key to decrypt the COSE object present in the payload so the `join_response` object is opaque to it. The response is matched to the request and verified for replay protection at pledge using OSCOAP processing rules. The response does not contain JRC's address as in this particular example, we assume that JRC is co-located with 6LBR.

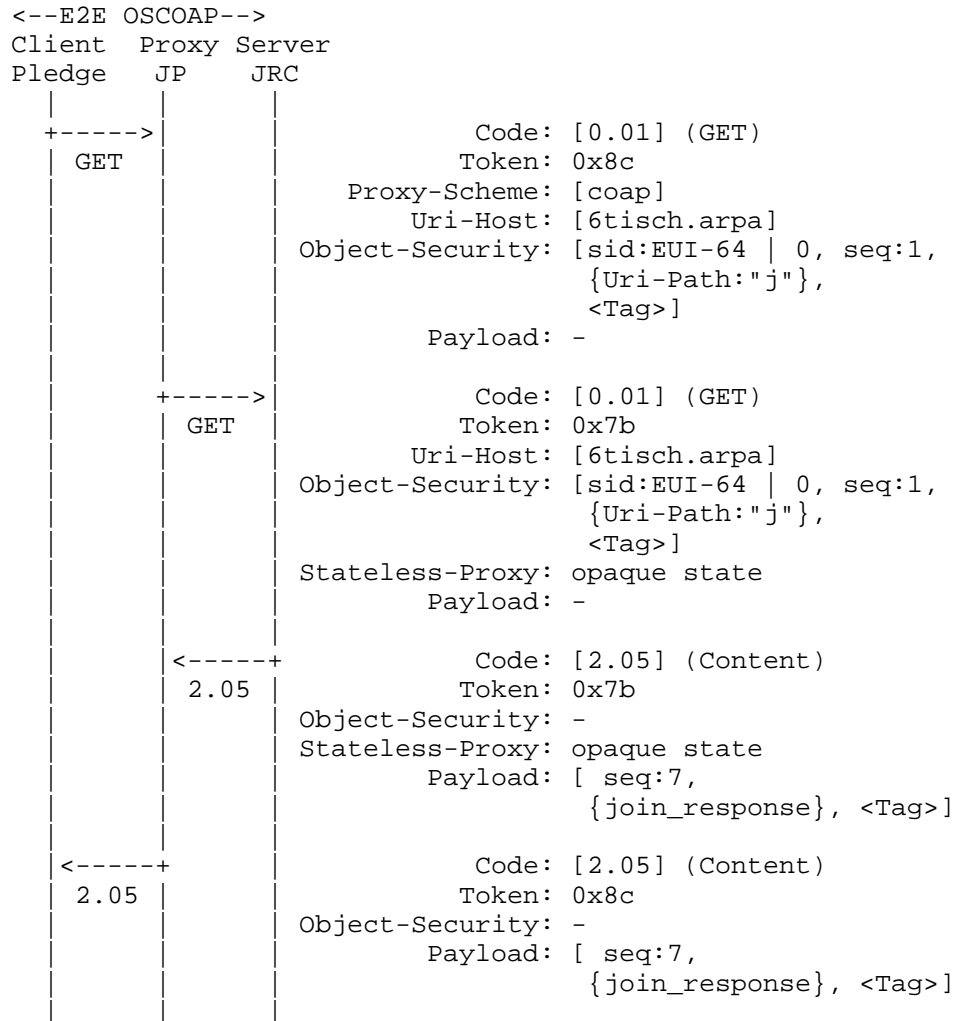


Figure 6: Example of a join protocol exchange with a PSK. {} denotes encryption and authentication, [] denotes authentication.

Where `join_response` is as follows.

```
join_response:
[
  [ / COSE Key Set array with a single key /
    {
      1 : 4, / key type symmetric /
      2 : h'01', / key id /
      -1 : h'e6bf4287c2d7618d6a9687445ffd33e6' / key value /
    }
  ],
  [
    h'af93' / assigned short address /
  ]
]
```

Encodes to
h'8281a301040241012050e6bf4287c2d7618d6a9687445ffd33e68142af93' with
a size of 30 bytes.

Authors' Addresses

Malisa Vucinic (editor)
Inria
2 Rue Simone Iff
Paris 75012
France

Email: malisa.vucinic@inria.fr

Jonathan Simon
Linear Technology
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: jsimon@linear.com

Kris Pister
University of California Berkeley
512 Cory Hall
Berkeley, CA 94720
USA

Email: pister@eecs.berkeley.edu

Michael Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z5V7
Canada

Email: mcr+ietf@sandelman.ca

6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 12, 2020

M. Vucinic, Ed.
Inria
J. Simon
Analog Devices
K. Pister
University of California Berkeley
M. Richardson
Sandelman Software Works
December 10, 2019

Constrained Join Protocol (CoJP) for 6TiSCH
draft-ietf-6tisch-minimal-security-15

Abstract

This document describes the minimal framework required for a new device, called "pledge", to securely join a 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) network. The framework requires that the pledge and the JRC (join registrar/coordinator, a central entity), share a symmetric key. How this key is provisioned is out of scope of this document. Through a single CoAP (Constrained Application Protocol) request-response exchange secured by OSCORE (Object Security for Constrained RESTful Environments), the pledge requests admission into the network and the JRC configures it with link-layer keying material and other parameters. The JRC may at any time update the parameters through another request-response exchange secured by OSCORE. This specification defines the Constrained Join Protocol and its CBOR (Concise Binary Object Representation) data structures, and describes how to configure the rest of the 6TiSCH communication stack for this join process to occur in a secure manner. Additional security mechanisms may be added on top of this minimal framework.

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 12, 2020.

Copyright Notice

Copyright (c) 2019 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	3
2. Terminology	4
3. Provisioning Phase	5
4. Join Process Overview	7
4.1. Step 1 - Enhanced Beacon	8
4.2. Step 2 - Neighbor Discovery	9
4.3. Step 3 - Constrained Join Protocol (CoJP) Execution	9
4.4. The Special Case of the 6LBR Pledge Joining	10
5. Link-layer Configuration	10
5.1. Distribution of Time	11
6. Network-layer Configuration	12
6.1. Identification of Unauthenticated Traffic	13
7. Application-level Configuration	14
7.1. Statelessness of the JP	15
7.2. Recommended Settings	16
7.3. OSCORE	16
8. Constrained Join Protocol (CoJP)	19
8.1. Join Exchange	20
8.2. Parameter Update Exchange	21
8.3. Error Handling	23
8.4. CoJP Objects	25
8.5. Recommended Settings	39
9. Security Considerations	39
10. Privacy Considerations	41
11. IANA Considerations	42
11.1. CoJP Parameters Registry	42
11.2. CoJP Key Usage Registry	43
11.3. CoJP Unsupported Configuration Code Registry	44
12. Acknowledgments	44

13. References	45
13.1. Normative References	45
13.2. Informative References	46
Appendix A. Example	48
Appendix B. Lightweight Implementation Option	51
Authors' Addresses	52

1. Introduction

This document defines a "secure join" solution for a new device, called "pledge", to securely join a 6TiSCH network. The term "secure join" refers to network access authentication, authorization and parameter distribution, as defined in [I-D.ietf-6tisch-architecture]. The Constrained Join Protocol (CoJP) defined in this document handles parameter distribution needed for a pledge to become a joined node. Mutual authentication during network access and implicit authorization are achieved through the use of a secure channel, as configured by this document. This document also specifies a configuration of different layers of the 6TiSCH protocol stack that reduces the Denial of Service (DoS) attack surface during the join process.

This document presumes a 6TiSCH network as described by [RFC7554] and [RFC8180]. By design, nodes in a 6TiSCH network [RFC7554] have their radio turned off most of the time, to conserve energy. As a consequence, the link used by a new device for joining the network has limited bandwidth [RFC8180]. The secure join solution defined in this document therefore keeps the number of over-the-air exchanges to a minimum.

The micro-controllers at the heart of 6TiSCH nodes have a small amount of code memory. It is therefore paramount to reuse existing protocols available as part of the 6TiSCH stack. At the application layer, the 6TiSCH stack already relies on CoAP [RFC7252] for web transfer, and on OSCORE [RFC8613] for its end-to-end security. The secure join solution defined in this document therefore reuses those two protocols as its building blocks.

CoJP is a generic protocol that can be used as-is in all modes of IEEE Std 802.15.4 [IEEE802.15.4], including the Time-Slotted Channel Hopping (TSCH) mode 6TiSCH is based on. CoJP may as well be used in other (low-power) networking technologies where efficiency in terms of communication overhead and code footprint is important. In such a case, it may be necessary to define configuration parameters specific to the technology in question, through companion documents. The overall process described in Section 4 and the configuration of the stack is specific to 6TiSCH.

CoJP assumes the presence of a Join Registrar/Coordinator (JRC), a central entity. The configuration defined in this document assumes that the pledge and the JRC share a unique symmetric cryptographic key, called PSK (pre-shared key). The PSK is used to configure OSCORE to provide a secure channel to CoJP. How the PSK is installed is out of scope of this document: this may happen during the provisioning phase or by a key exchange protocol that may precede the execution of CoJP.

When the pledge seeks admission to a 6TiSCH network, it first synchronizes to it, by initiating the passive scan defined in [IEEE802.15.4]. The pledge then exchanges CoJP messages with the JRC; for this end-to-end communication to happen, messages are forwarded by nodes already part of the 6TiSCH network, called Join Proxies. The messages exchanged allow the JRC and the pledge to mutually authenticate, based on the properties provided by OSCORE. They also allow the JRC to configure the pledge with link-layer keying material, short identifier and other parameters. After this secure join process successfully completes, the joined node can interact with its neighbors to request additional bandwidth using the 6top Protocol [RFC8480] and start sending application traffic.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The reader is expected to be familiar with the terms and concepts defined in [I-D.ietf-6tisch-architecture], [RFC7252], [RFC8613], and [RFC8152].

The specification also includes a set of informative specifications using the Concise data definition language (CDDL) [I-D.ietf-cbor-cddl].

The following terms defined in [I-D.ietf-6tisch-architecture] are used extensively throughout this document:

- o pledge
- o joined node
- o join proxy (JP)
- o join registrar/coordinator (JRC)

- o enhanced beacon (EB)
- o join protocol
- o join process

The following terms defined in [RFC8505] are also used throughout this document:

- o 6LoWPAN Border Router (6LBR)
- o 6LoWPAN Node (6LN)

The term "6LBR" is used interchangeably with the term "DODAG root" defined in [RFC6550], on the assumption that the two entities are co-located, as recommended by [I-D.ietf-6tisch-architecture].

The term "pledge", as used throughout the document, explicitly denotes non-6LBR devices attempting to join the network using their IEEE Std 802.15.4 network interface. The device that attempts to join as the 6LBR of the network and does so over another network interface is explicitly denoted as the "6LBR pledge". When the text equally applies to the pledge and the 6LBR pledge, the "(6LBR) pledge" form is used.

In addition, we use generic terms "pledge identifier" and "network identifier". See Section 3.

3. Provisioning Phase

The (6LBR) pledge is provisioned with certain parameters before attempting to join the network, and the same parameters are provisioned to the JRC. There are many ways by which this provisioning can be done. Physically, the parameters can be written into the (6LBR) pledge using a number of mechanisms, such as a JTAG interface, a serial (craft) console interface, pushing buttons simultaneously on different devices, over-the-air configuration in a Faraday cage, etc. The provisioning can be done by the vendor, the manufacturer, the integrator, etc.

Details of how this provisioning is done is out of scope of this document. What is assumed is that there can be a secure, private conversation between the JRC and the (6LBR) pledge, and that the two devices can exchange the parameters.

Parameters that are provisioned to the (6LBR) pledge include:

- o pledge identifier. The pledge identifier identifies the (6LBR) pledge. The pledge identifier MUST be unique in the set of all pledge identifiers managed by a JRC. The pledge identifier uniqueness is an important security requirement, as discussed in Section 9. The pledge identifier is typically the globally unique 64-bit Extended Unique Identifier (EUI-64) of the IEEE Std 802.15.4 device, in which case it is provisioned by the hardware manufacturer. The pledge identifier is used to generate the IPv6 addresses of the (6LBR) pledge and to identify it during the execution of the join protocol. Depending on the configuration, the pledge identifier may also be used after the join process to identify the joined node. For privacy reasons (see Section 10), it is possible to use a pledge identifier different from the EUI-64. For example, a pledge identifier may be a random byte string, but care needs to be taken that such a string meets the uniqueness requirement.
- o Pre-Shared Key (PSK). A symmetric cryptographic key shared between the (6LBR) pledge and the JRC. To look up the PSK for a given pledge, the JRC additionally needs to store the corresponding pledge identifier. Each (6LBR) pledge MUST be provisioned with a unique PSK. The PSK MUST be a cryptographically strong key, with at least 128 bits of entropy, indistinguishable by feasible computation from a random uniform string of the same length. How the PSK is generated and/or provisioned is out of scope of this specification. This could be done during a provisioning step or companion documents can specify the use of a key agreement protocol. Common pitfalls when generating PSKs are discussed in Section 9. In case of device re-commissioning to a new owner, the PSK MUST be changed. Note that the PSK is different from the link-layer keys K1 and K2 specified in [RFC8180]. The PSK is a long-term secret used to protect the execution of the secure join protocol specified in this document whose one output are link-layer keys.
- o Optionally, a network identifier. The network identifier identifies the 6TiSCH network. The network identifier MUST be carried within Enhanced Beacon (EB) frames. Typically, the 16-bit Personal Area Network Identifier (PAN ID) defined in [IEEE802.15.4] is used as the network identifier. However, PAN ID is not considered a stable network identifier as it may change during network lifetime if a collision with another network is detected. Companion documents can specify the use of a different network identifier for join purposes, but this is out of scope of this specification. Provisioning the network identifier to a pledge is RECOMMENDED. However, due to operational constraints, the network identifier may not be known at the time when the provisioning is done. In case this parameter is not provisioned

to the pledge, the pledge attempts to join one advertised network at a time, which significantly prolongs the join process. This parameter MUST be provisioned to the 6LBR pledge.

- o Optionally, any non-default algorithms. The default algorithms are specified in Section 7.3.3. When algorithm identifiers are not provisioned, the use of these default algorithms is implied.

Additionally, the 6LBR pledge that is not co-located with the JRC needs to be provisioned with:

- o Global IPv6 address of the JRC. This address is used by the 6LBR pledge to address the JRC during the join process. The 6LBR pledge may also obtain the IPv6 address of the JRC through other available mechanisms, such as DHCPv6 [RFC8415], GRASP [I-D.ietf-anima-grasp], mDNS [RFC6762], the use of which is out of scope of this document. Pledges do not need to be provisioned with this address as they discover it dynamically through CoJP.

4. Join Process Overview

This section describes the steps taken by a pledge in a 6TiSCH network. When a pledge seeks admission to a 6TiSCH network, the following exchange occurs:

1. The pledge listens for an Enhanced Beacon (EB) frame [IEEE802.15.4]. This frame provides network synchronization information, and tells the device when it can send a frame to the node sending the beacons, which acts as a Join Proxy (JP) for the pledge, and when it can expect to receive a frame. The Enhanced Beacon provides the link-layer address of the JP and it may also provide its link-local IPv6 address.
2. The pledge configures its link-local IPv6 address and advertises it to the JP using Neighbor Discovery. The advertisement step may be omitted if the link-local address has been derived from a known unique interface identifier, such as an EUI-64 address.
3. The pledge sends a Join Request to the JP in order to securely identify itself to the network. The Join Request is forwarded to the JRC.
4. In case of successful processing of the request, the pledge receives a Join Response from the JRC (via the JP). The Join Response contains configuration parameters necessary for the pledge to join the network.

From the pledge's perspective, joining is a local phenomenon – the pledge only interacts with the JP, and it needs not know how far it is from the 6LBR, or how to route to the JRC. Only after establishing one or more link-layer keys does it need to know about the particulars of a 6TiSCH network.

The join process is shown as a transaction diagram in Figure 1:

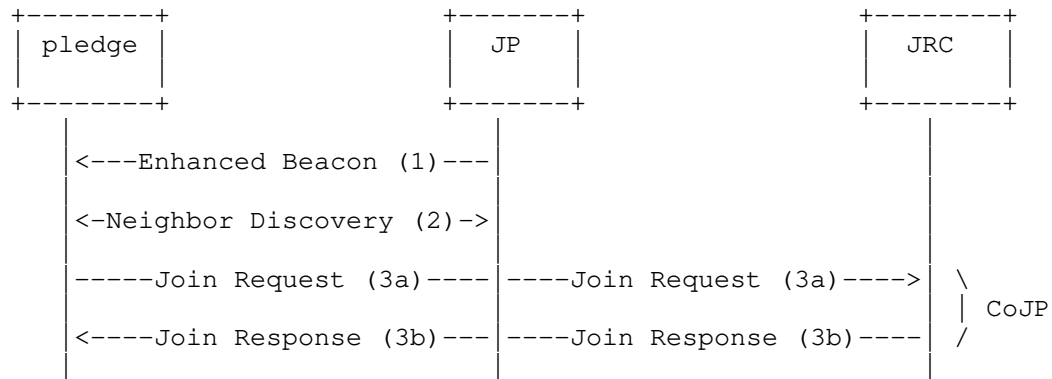


Figure 1: Overview of a successful join process.

As for other nodes in the network, the 6LBR node may act as the JP. The 6LBR may in addition be co-located with the JRC.

The details of each step are described in the following sections.

4.1. Step 1 – Enhanced Beacon

The pledge synchronizes to the network by listening for, and receiving, an Enhanced Beacon (EB) sent by a node already in the network. This process is entirely defined by [IEEE802.15.4], and described in [RFC7554].

Once the pledge hears an EB, it synchronizes to the joining schedule using the cells contained in the EB. The pledge can hear multiple EBs; the selection of which EB to use is out of the scope for this document, and is discussed in [RFC7554]. Implementers should make use of information such as: what network identifier the EB contains, the value of the Join Metric field within EBs, whether the source link-layer address of the EB has been tried before, what signal strength the different EBs were received at, etc. In addition, the pledge may be pre-configured to search for EBs with a specific network identifier.

If the pledge is not provisioned with the network identifier, it attempts to join one network at a time, as described in Section 8.1.1.

Once the pledge selects the EB, it synchronizes to it and transitions into a low-power mode. It follows the schedule information contained in the EB which indicates the slots that the pledge may use for the join process. During the remainder of the join process, the node that has sent the EB to the pledge acts as the JP.

At this point, the pledge may proceed to step 2, or continue to listen for additional EBs.

4.2. Step 2 - Neighbor Discovery

The pledge forms its link-local IPv6 address based on the interface identifier, as per [RFC4944]. The pledge MAY perform the Neighbor Solicitation / Neighbor Advertisement exchange with the JP, as per Section 5.6 of [RFC8505]. As per [RFC8505], there is no need to perform duplicate address detection for the link-local address. The pledge and the JP use their link-local IPv6 addresses for all subsequent communication during the join process.

Note that Neighbor Discovery exchanges at this point are not protected with link-layer security as the pledge is not in possession of the keys. How the JP accepts these unprotected frames is discussed in Section 5.

4.3. Step 3 - Constrained Join Protocol (CoJP) Execution

The pledge triggers the join exchange of the Constrained Join Protocol (CoJP). The join exchange consists of two messages: the Join Request message (Step 3a), and the Join Response message conditioned on the successful security processing of the request (Step 3b).

All CoJP messages are exchanged over a secure end-to-end channel that provides confidentiality, data authenticity and replay protection. Frames carrying CoJP messages are not protected with link-layer security when exchanged between the pledge and the JP as the pledge is not in possession of the link-layer keys in use. How JP and pledge accept these unprotected frames is discussed in Section 5. When frames carrying CoJP messages are exchanged between nodes that have already joined the network, the link-layer security is applied according to the security configuration used in the network.

4.3.1. Step 3a - Join Request

The Join Request is a message sent from the pledge to the JP, and which the JP forwards to the JRC. The pledge indicates in the Join Request the role it requests to play in the network, as well as the identifier of the network it requests to join. The JP forwards the Join Request to the JRC on the existing links. How exactly this happens is out of scope of this document; some networks may wish to dedicate specific link layer resources for this join traffic.

4.3.2. Step 3b - Join Response

The Join Response is sent by the JRC to the pledge, and is forwarded through the JP. The packet containing the Join Response travels from the JRC to the JP using the operating routes in the network. The JP delivers it to the pledge. The JP operates as an application-layer proxy, see Section 7.

The Join Response contains different parameters needed by the pledge to become a fully operational network node. These parameters include the link-layer key(s) currently in use in the network, the short address assigned to the pledge, the IPv6 address of the JRC needed by the pledge to operate as the JP, among others.

4.4. The Special Case of the 6LBR Pledge Joining

The 6LBR pledge performs Section 4.3 of the join process described above, just as any other pledge, albeit over a different network interface. There is no JP intermediating the communication between the 6LBR pledge and the JRC, as described in Section 6. The other steps of the described join process do not apply to the 6LBR pledge. How the 6LBR pledge obtains an IPv6 address and triggers the execution of the CoJP protocol is out of scope of this document.

5. Link-layer Configuration

In an operational 6TiSCH network, all frames use link-layer frame security [RFC8180]. The IEEE Std 802.15.4 security attributes include frame authenticity, and optionally frame confidentiality (i.e. encryption).

Any node sending EB frames MUST be prepared to act as a JP for potential pledges.

The pledge does not initially do any authenticity check of the EB frames, as it does not possess the link-layer key(s) in use. The pledge is still able to parse the contents of the received EBs and synchronize to the network, as EBs are not encrypted [RFC8180].

When sending frames during the join process, the pledge sends unencrypted and unauthenticated frames at the link layer. In order for the join process to be possible, the JP must accept these unsecured frames for the duration of the join process. This behavior may be implemented by setting the "secExempt" attribute in the IEEE Std 802.15.4 security configuration tables. It is expected that the lower layer provides an interface to indicate to the upper layer that unsecured frames are being received from a device, and that the upper layer can use that information to make a determination that a join process is in place and the unsecured frames should be processed. How the JP makes such a determination and interacts with the lower layer is out of scope of this specification. The JP can additionally make use of information such as the value of the join rate parameter (Section 8.4.2) set by the JRC, physical button press, etc.

When the pledge initially synchronizes to the network, it has no means of verifying the authenticity of EB frames. As an attacker can craft a frame that looks like a legitimate EB frame this opens up a DoS vector, as discussed in Section 9.

5.1. Distribution of Time

Nodes in a 6TiSCH network keep a global notion of time known as the absolute slot number. Absolute slot number is used in the construction of the link-layer nonce, as defined in [IEEE802.15.4]. The pledge initially synchronizes to the EB frame sent by the JP, and uses the value of the absolute slot number found in the TSCH Synchronization Information Element. At the time of the synchronization, the EB frame can neither be authenticated nor its freshness verified. During the join process, the pledge sends frames that are unprotected at the link-layer and protected end-to-end instead. The pledge does not obtain the time information as the output of the join process as this information is local to the network and may not be known at the JRC.

This enables an attack on the pledge where the attacker replays to the pledge legitimate EB frames obtained from the network and acts as a man-in-the-middle between the pledge and the JP. The EB frames will make the pledge believe that the replayed absolute slot number value is the current notion of time in the network. By forwarding the join traffic to the legitimate JP, the attacker enables the pledge to join the network. Under different conditions relating to the reuse of the pledge's short address by the JRC or its attempt to rejoin the network, this may cause the pledge to reuse the link-layer nonce in the first frame it sends protected after the join process is completed.

For this reason, all frames originated at the JP and destined to the pledge during the join process MUST be authenticated at the link-layer using the key that is normally in use in the network. Link-layer security processing at the pledge for these frames will fail as the pledge is not yet in possession of the key. The pledge acknowledges these frames without link-layer security, and JP accepts the unsecured acknowledgment due to the secExempt attribute set for the pledge. The frames should be passed to the upper layer for processing using the promiscuous mode of [IEEE802.15.4] or another appropriate mechanism. When the upper layer processing on the pledge is completed and the link-layer keys are configured, the upper layer MUST trigger the security processing of the corresponding frame. Once the security processing of the frame carrying the Join Response message is successful, the current absolute slot number kept locally at the pledge SHALL be declared as valid.

6. Network-layer Configuration

The pledge and the JP SHOULD keep a separate neighbor cache for untrusted entries and use it to store each other's information during the join process. Mixing neighbor entries belonging to pledges and nodes that are part of the network opens up the JP to a DoS attack, as the attacker may fill JP's neighbor table and prevent the discovery of legitimate neighbors.

Once the pledge obtains link-layer keys and becomes a joined node, it is able to securely communicate with its neighbors, obtain the network IPv6 prefix and form its global IPv6 address. The joined node then undergoes an independent process to bootstrap its neighbor cache entries, possibly with a node that formerly acted as a JP, following [RFC8505]. From the point of view of the JP, there is no relationship between the neighbor cache entry belonging to a pledge and the joined node that formerly acted as a pledge.

The pledge does not communicate with the JRC at the network layer. This allows the pledge to join without knowing the IPv6 address of the JRC. Instead, the pledge communicates with the JP at the network layer using link-local addressing, and with the JRC at the application layer, as specified in Section 7.

The JP communicates with the JRC over global IPv6 addresses. The JP discovers the network IPv6 prefix and configures its global IPv6 address upon successful completion of the join process and the obtention of link-layer keys. The pledge learns the IPv6 address of the JRC from the Join Response, as specified in Section 8.1.2; it uses it once joined in order to operate as a JP.

As a special case, the 6LBR pledge may have an additional network interface that it uses in order to obtain the configuration parameters from the JRC and start advertising the 6TiSCH network. This additional interface needs to be configured with a global IPv6 address, by a mechanism that is out of scope of this document. The 6LBR pledge uses this interface to directly communicate with the JRC using global IPv6 addressing.

The JRC can be co-located on the 6LBR. In this special case, the IPv6 address of the JRC can be omitted from the Join Response message for space optimization. The 6LBR then MUST set the DODAGID field in the RPL DIOs [RFC6550] to its IPv6 address. The pledge learns the address of the JRC once joined and upon the reception of the first RPL DIO message, and uses it to operate as a JP.

6.1. Identification of Unauthenticated Traffic

The traffic that is proxied by the Join Proxy (JP) comes from unauthenticated pledges, and there may be an arbitrary amount of it. In particular, an attacker may send fraudulent traffic in an attempt to overwhelm the network.

When operating as part of a [RFC8180] 6TiSCH minimal network using distributed scheduling algorithms, the traffic from unauthenticated pledges may cause intermediate nodes to request additional bandwidth. An attacker could use this property to cause the network to overcommit bandwidth (and energy) to the join process.

The Join Proxy is aware of what traffic originates from unauthenticated pledges, and so can avoid allocating additional bandwidth itself. The Join Proxy implements a data cap on outgoing join traffic by implementing the recommendation of 1 packet per 3 seconds in Section 3.1.3 of [RFC8085]. This can be achieved with the congestion control mechanism specified in Section 4.7 of [RFC7252]. This cap will not protect intermediate nodes as they can not tell join traffic from regular traffic. Despite the data cap implemented separately on each Join Proxy, the aggregate join traffic from many Join Proxies may cause intermediate nodes to decide to allocate additional cells. It is undesirable to do so in response to the traffic originated at unauthenticated pledges. In order to permit the intermediate nodes to avoid this, the traffic needs to be tagged. [RFC2597] defines a set of per-hop behaviors that may be encoded into the Diffserv Code Points (DSCPs). Based on the DSCP, intermediate nodes can decide whether to act on a given packet.

6.1.1.1. Traffic from JP to JRC

The Join Proxy SHOULD set the DSCP of packets that it produces as part of the forwarding process to AF43 code point (See Section 6 of [RFC2597]). A Join Proxy that does not require a specific DSCP value on traffic forwarded should set it to zero so that it is compressed out.

A Scheduling Function (SF) running on 6TiSCH nodes SHOULD NOT allocate additional cells as a result of traffic with code point AF43. Companion SF documents SHOULD specify how this recommended behavior is achieved. One example is the 6TiSCH Minimal Scheduling Function [I-D.ietf-6tisch-msf].

6.1.1.2. Traffic from JRC to JP

The JRC SHOULD set the DSCP of join response packets addressed to the Join Proxy to AF42 code point. AF42 has lower drop probability than AF43, giving this traffic priority in buffers over the traffic going towards the JRC.

The 6LBR links are often the most congested within a DODAG, and from that point down there is progressively less (or equal) congestion. If the 6LBR paces itself when sending join response traffic then it ought to never exceed the bandwidth allocated to the best effort traffic cells. If the 6LBR has the capacity (if it is not constrained) then it should provide some buffers in order to satisfy the Assured Forwarding behavior.

Companion SF documents SHOULD specify how traffic with code point AF42 is handled with respect to cell allocation. In case the recommended behavior described in this section is not followed, the network may become prone to the attack discussed in Section 6.1.

7. Application-level Configuration

The CoJP join exchange in Figure 1 is carried over CoAP [RFC7252] and the secure channel provided by OSCORE [RFC8613]. The (6LBR) pledge acts as a CoAP client; the JRC acts as a CoAP server. The JP implements CoAP forward proxy functionality [RFC7252]. Because the JP can also be a constrained device, it cannot implement a cache.

The pledge designates a JP as a proxy by including the Proxy-Scheme option in CoAP requests it sends to the JP. The pledge also includes in the requests the Uri-Host option with its value set to the well-known JRC's alias, as specified in Section 8.1.1.

The JP resolves the alias to the IPv6 address of the JRC that it learned when it acted as a pledge, and joined the network. This allows the JP to reach the JRC at the network layer and forward the requests on behalf of the pledge.

7.1. Statelessness of the JP

The CoAP proxy defined in [RFC7252] keeps per-client state information in order to forward the response towards the originator of the request. This state information includes at least the CoAP token, the IPv6 address of the client, and the UDP source port number. Since the JP can be a constrained device that acts as a CoAP proxy, memory limitations make it prone to a Denial-of-Service (DoS) attack.

This DoS vector on the JP can be mitigated by making the JP act as a stateless CoAP proxy, where "state" encompasses the information related to individual pledges. The JP can wrap the state it needs to keep for a given pledge throughout the network stack in a "state object" and include it as a CoAP token in the forwarded request to the JRC. The JP may use the CoAP token as defined in [RFC7252], if the size of the serialized state object permits, or use the extended CoAP token defined in [I-D.ietf-core-stateless], to transport the state object. The JRC and any other potential proxy on the JP - JRC path MUST support extended token lengths, as defined in [I-D.ietf-core-stateless]. Since the CoAP token is echoed back in the response, the JP is able to decode the state object and configure the state needed to forward the response to the pledge. The information that the JP needs to encode in the state object to operate in a fully stateless manner with respect to a given pledge is implementation specific.

It is RECOMMENDED that the JP operates in a stateless manner and signals the per-pledge state within the CoAP token, for every request it forwards into the network on behalf of unauthenticated pledges. When the JP is operating in a stateless manner, the security considerations from [I-D.ietf-core-stateless] apply and the type of the CoAP message that the JP forwards on behalf of the pledge MUST be non-confirmable (NON), regardless of the message type received from the pledge. The use of a non-confirmable message by the JP alleviates the JP from keeping CoAP message exchange state. The retransmission burden is then entirely shifted to the pledge. A JP that operates in a stateless manner still needs to keep congestion control state with the JRC, see Section 9. Recommended values of CoAP settings for use during the join process, both by the pledge and the JP, are given in Section 7.2.

Note that in some networking stack implementations, a fully (per-pledge) stateless operation of the JP may be challenging from the implementation's point of view. In those cases, the JP may operate as a statefull proxy that stores the per-pledge state until the response is received or timed out, but this comes at a price of a DoS vector.

7.2. Recommended Settings

This section gives RECOMMENDED values of CoAP settings during the join process.

Name	Default Value
ACK_TIMEOUT	10 seconds
ACK_RANDOM_FACTOR	1.5
MAX_RETRANSMIT	4
NSTART	1
DEFAULT_LEISURE	5 seconds
PROBING_RATE	1 byte/second

Recommended CoAP settings.

These values may be configured to values specific to the deployment. The default values have been chosen to accommodate a wide range of deployments, taking into account dense networks.

The PROBING_RATE value at the JP is controlled by the join rate parameter, see Section 8.4.2. Following [RFC7252], the average data rate in sending to the JRC must not exceed PROBING_RATE. For security reasons, the average data rate SHOULD be measured over a rather short window, e.g. ACK_TIMEOUT, see Section 9.

7.3. OSCORE

Before the (6LBR) pledge and the JRC start exchanging CoAP messages protected with OSCORE, they need to derive the OSCORE security context from the provisioned parameters, as discussed in Section 3.

The OSCORE security context MUST be derived as per Section 3 of [RFC8613].

- o the Master Secret MUST be the PSK.
- o the Master Salt MUST be the empty byte string.
- o the ID Context MUST be set to the pledge identifier.
- o the ID of the pledge MUST be set to the empty byte string. This identifier is used as the OSCORE Sender ID of the pledge in the security context derivation, since the pledge initially acts as a CoAP client.
- o the ID of the JRC MUST be set to the byte string 0x4a5243 ("JRC" in ASCII). This identifier is used as the OSCORE Recipient ID of the pledge in the security context derivation, as the JRC initially acts as a CoAP server.
- o the Algorithm MUST be set to the value from [RFC8152], agreed out-of-band by the same mechanism used to provision the PSK. The default is AES-CCM-16-64-128.
- o the Key Derivation Function MUST be agreed out-of-band by the same mechanism used to provision the PSK. Default is HKDF SHA-256 [RFC5869].

Since the pledge's OSCORE Sender ID is the empty byte string, when constructing the OSCORE option, the pledge sets the k bit in the OSCORE flag byte, but indicates a 0-length kid. The pledge transports its pledge identifier within the kid context field of the OSCORE option. The derivation in [RFC8613] results in OSCORE keys and a common IV for each side of the conversation. Nonces are constructed by XOR'ing the common IV with the current sequence number. For details on nonce and OSCORE option construction, refer to [RFC8613].

Implementations MUST ensure that multiple CoAP requests, including to different JRCs, are properly incrementing the sequence numbers, so that the same sequence number is never reused in distinct requests protected under the same PSK. The pledge typically sends requests to different JRCs if it is not provisioned with the network identifier and attempts to join one network at a time. Failure to comply will break the security guarantees of the Authenticated Encryption with Associated Data (AEAD) algorithm because of nonce reuse.

This OSCORE security context is used for initial joining of the (6LBR) pledge, where the (6LBR) pledge acts as a CoAP client, as well as for any later parameter updates, where the JRC acts as a CoAP client and the joined node as a CoAP server, as discussed in Section 8.2. Note that when the (6LBR) pledge and the JRC change

roles between CoAP client and CoAP server, the same OSCORE security context as initially derived remains in use and the derived parameters are unchanged, for example Sender ID when sending and Recipient ID when receiving (see Section 3.1 of [RFC8613]). A (6LBR) pledge is expected to have exactly one OSCORE security context with the JRC.

7.3.1. Replay Window and Persistency

Both (6LBR) pledge and the JRC MUST implement a replay protection mechanism. The use of the default OSCORE replay protection mechanism specified in Section 3.2.2 of [RFC8613] is RECOMMENDED.

Implementations MUST ensure that mutable OSCORE context parameters (Sender Sequence Number, Replay Window) are stored in persistent memory. A technique detailed in Appendix B.1.1 of [RFC8613] that prevents reuse of sequence numbers MUST be implemented. Each update of the OSCORE Replay Window MUST be written to persistent memory.

This is an important security requirement in order to guarantee nonce uniqueness and resistance to replay attacks across reboots and rejoins. Traffic between the (6LBR) pledge and the JRC is rare, making security outweigh the cost of writing to persistent memory.

7.3.2. OSCORE Error Handling

Errors raised by OSCORE during the join process MUST be silently dropped, with no error response being signaled. The pledge MUST silently discard any response not protected with OSCORE, including error codes.

Such errors may happen for a number of reasons, including failed lookup of an appropriate security context (e.g. the pledge attempting to join a wrong network), failed decryption, positive replay window lookup, formatting errors (possibly due to malicious alterations in transit). Silently dropping OSCORE messages prevents a DoS attack on the pledge where the attacker could send bogus error responses, forcing the pledge to attempt joining one network at a time, until all networks have been tried.

7.3.3. Mandatory to Implement Algorithms

The mandatory to implement AEAD algorithm for use with OSCORE is AES-CCM-16-64-128 from [RFC8152]. This is the algorithm used for securing IEEE Std 802.15.4 frames, and hardware acceleration for it is present in virtually all compliant radio chips. With this choice, CoAP messages are protected with an 8-byte CCM authentication tag, and the algorithm uses 13-byte long nonces.

The mandatory to implement hash algorithm is SHA-256 [RFC4231]. The mandatory to implement key derivation function is HKDF [RFC5869], instantiated with a SHA-256 hash. See Appendix B for implementation guidance when code footprint is important.

8. Constrained Join Protocol (CoJP)

The Constrained Join Protocol (CoJP) is a lightweight protocol over CoAP [RFC7252] and a secure channel provided by OSCORE [RFC8613]. CoJP allows a (6LBR) pledge to request admission into a network managed by the JRC. It enables the JRC to configure the pledge with the necessary parameters. The JRC may update the parameters at any time, by reaching out to the joined node that formerly acted as a (6LBR) pledge. For example, network-wide rekeying can be implemented by updating the keying material on each node.

CoJP relies on the security properties provided by OSCORE. This includes end-to-end confidentiality, data authenticity, replay protection, and a secure binding of responses to requests.

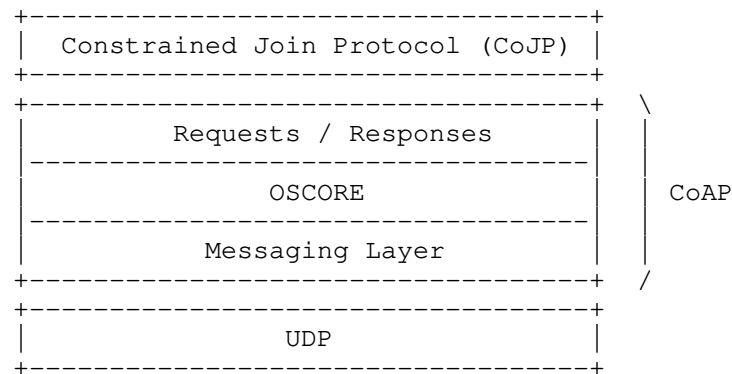


Figure 2: Abstract layering of CoJP.

When a (6LBR) pledge requests admission to a given network, it undergoes the CoJP join exchange that consists of:

- o the Join Request message, sent by the (6LBR) pledge to the JRC, potentially proxied by the JP. The Join Request message and its mapping to CoAP is specified in Section 8.1.1.
- o the Join Response message, sent by the JRC to the (6LBR) pledge, if the JRC successfully processes the Join Request using OSCORE and it determines through a mechanism that is out of scope of this specification that the (6LBR) pledge is authorized to join the network. The Join Response message is potentially proxied by the

JP. The Join Response message and its mapping to CoAP is specified in Section 8.1.2.

When the JRC needs to update the parameters of a joined node that formerly acted as a (6LBR) pledge, it executes the CoJP parameter update exchange that consists of:

- o the Parameter Update message, sent by the JRC to the joined node that formerly acted as a (6LBR) pledge. The Parameter Update message and its mapping to CoAP is specified in Section 8.2.1.

The payload of CoJP messages is encoded with CBOR [RFC7049]. The CBOR data structures that may appear as the payload of different CoJP messages are specified in Section 8.4.

8.1. Join Exchange

This section specifies the messages exchanged when the (6LBR) pledge requests admission and configuration parameters from the JRC.

8.1.1. Join Request Message

The Join Request message that the (6LBR) pledge sends SHALL be mapped to a CoAP request:

- o The request method is POST.
- o The type is Confirmable (CON).
- o The Proxy-Scheme option is set to "coap".
- o The Uri-Host option is set to "6tisch.arpa". This is an anycast type of identifier of the JRC that is resolved to its IPv6 address by the JP or the 6LBR pledge.
- o The Uri-Path option is set to "j".
- o The OSCORE option SHALL be set according to [RFC8613]. The OSCORE security context used is the one derived in Section 7.3. The OSCORE kid context allows the JRC to retrieve the security context for a given pledge.
- o The payload is a Join_Request CBOR object, as defined in Section 8.4.1.

Since the Join Request is a confirmable message, the transmission at (6LBR) pledge will be controlled by CoAP's retransmission mechanism. The JP, when operating in a stateless manner, forwards this Join

Request as a non-confirmable (NON) CoAP message, as specified in Section 7. If the CoAP implementation at (6LBR) pledge declares the message transmission as failure, the (6LBR) pledge SHOULD attempt to join a 6TiSCH network advertised with a different network identifier. See Section 7.2 for recommended values of CoAP settings to use during the join exchange.

If all join attempts to advertised networks have failed, the (6LBR) pledge SHOULD signal the presence of an error condition, through some out-of-band mechanism.

BCP190 [RFC7320] provides guidelines on URI design and ownership. It recommends that whenever a third party wants to mandate a URL to web authority that it SHOULD go under `"/.well-known"` (as per [RFC5785]). In the case of CoJP, the Uri-Host option is always set to `"6tisch.arpa"`, and based upon the recommendations in the Introduction of [RFC7320], it is asserted that this document is the owner of the CoJP service. As such, the concerns of [RFC7320] do not apply, and thus the Uri-Path is only `"/j"`.

8.1.2. Join Response Message

The Join Response message that the JRC sends SHALL be mapped to a CoAP response:

- o The response Code is 2.04 (Changed).
- o The payload is a Configuration CBOR object, as defined in Section 8.4.2.

8.2. Parameter Update Exchange

During the network lifetime, parameters returned as part of the Join Response may need to be updated. One typical example is the update of link-layer keying material for the network, a process known as rekeying. This section specifies a generic mechanism when this parameter update is initiated by the JRC.

At the time of the join, the (6LBR) pledge acts as a CoAP client and requests the network parameters through a representation of the `"/j"` resource, exposed by the JRC. In order for the update of these parameters to happen, the JRC needs to asynchronously contact the joined node. The use of the CoAP Observe option for this purpose is not feasible due to the change in the IPv6 address when the pledge becomes the joined node and obtains a global address.

Instead, once the (6LBR) pledge receives and successfully validates the Join Response and so becomes a joined node, it becomes a CoAP

server. The joined node creates a CoAP service at the Uri-Host value of "6tisch.arpa", and the joined node exposes the "/" resource that is used by the JRC to update the parameters. Consequently, the JRC operates as a CoAP client when updating the parameters. The request/response exchange between the JRC and the (6LBR) pledge happens over the already-established OSCORE secure channel.

8.2.1. Parameter Update Message

The Parameter Update message that the JRC sends to the joined node SHALL be mapped to a CoAP request:

- o The request method is POST.
- o The type is Confirmable (CON).
- o The Uri-Host option is set to "6tisch.arpa".
- o The Uri-Path option is set to "/".
- o The OSCORE option SHALL be set according to [RFC8613]. The OSCORE security context used is the one derived in Section 7.3. When a joined node receives a request with the Sender ID set to 0x4a5243 (ID of the JRC), it is able to correctly retrieve the security context with the JRC.
- o The payload is a Configuration CBOR object, as defined in Section 8.4.2.

The JRC has implicit knowledge on the global IPv6 address of the joined node, as it knows the pledge identifier that the joined node used when it acted as a pledge, and the IPv6 network prefix. The JRC uses this implicitly derived IPv6 address of the joined node to directly address CoAP messages to it.

In case the JRC does not receive a response to a Parameter Update message, it attempts multiple retransmissions, as configured by the underlying CoAP retransmission mechanism triggered for confirmable messages. Finally, if the CoAP implementation declares the transmission as failure, the JRC may consider this as a hint that the joined node is no longer in the network. How the JRC decides when to stop attempting to contact a previously joined node is out of scope of this specification but security considerations on the reuse of assigned resources apply, as discussed in Section 9.

8.3. Error Handling

8.3.1. CoJP CBOR Object Processing

CoJP CBOR objects are transported within both CoAP requests and responses. This section describes handling in case certain CoJP CBOR object parameters are not supported by the implementation or their processing fails. See Section 7.3.2 for the handling of errors that may be raised by the underlying OSCORE implementation.

When such a parameter is detected in a CoAP request (Join Request message, Parameter Update message), a Diagnostic Response message MUST be returned. A Diagnostic Response message maps to a CoAP response and is specified in Section 8.3.2.

When a parameter that cannot be acted upon is encountered while processing a CoJP object in a CoAP response (Join Response message), a (6LBR) pledge SHOULD attempt to join. In this case, the (6LBR) pledge SHOULD include the Unsupported Configuration CBOR object within the Join Request object in the following Join Request message. The Unsupported Configuration CBOR object is self-contained and enables the (6LBR) pledge to signal any parameters that the implementation of the networking stack may not support. A (6LBR) pledge MUST NOT attempt more than COJP_MAX_JOIN_ATTEMPTS number of attempts to join if the processing of the Join Response message fails each time. If COJP_MAX_JOIN_ATTEMPTS number of attempts is reached without success, the (6LBR) pledge SHOULD signal the presence of an error condition, through some out-of-band mechanism.

Note that COJP_MAX_JOIN_ATTEMPTS relates to the application-level handling of the CoAP response and is different from CoAP's MAX_RETRANSMIT setting that drives the retransmission mechanism of the underlying CoAP message.

8.3.2. Diagnostic Response Message

The Diagnostic Response message is returned for any CoJP request when the processing of the payload failed. The Diagnostic Response message is protected by OSCORE as any other CoJP protocol message.

The Diagnostic Response message SHALL be mapped to a CoAP response:

- o The response Code is 4.00 (Bad Request).
- o The payload is an Unsupported Configuration CBOR object, as defined in Section 8.4.5, containing more information about the parameter that triggered the sending of this message.

8.3.3. Failure Handling

The Parameter Update exchange may be triggered at any time during the network lifetime, which may span several years. During this period, it may occur that a joined node or the JRC experience unexpected events such as reboots or complete failures.

This document mandates that the mutable parameters in the security context are written to persistent memory (see Section 7.3.1) by both the JRC and pledges (joined nodes). As the joined node (pledge) is typically a constrained device that handles the write operations to persistent memory in a predictable manner, the retrieval of mutable security context parameters is feasible across reboots such that there is no risk of AEAD nonce reuse due to reinitialized Sender Sequence numbers, or of a replay attack due to the reinitialized replay window. JRC may be hosted on a generic machine where the write operation to persistent memory may lead to unpredictable delays due to caching. In case of a reboot event at JRC occurring before the cached data is written to persistent memory, the loss of mutable security context parameters is likely which consequently poses the risk of AEAD nonce reuse.

In the event of a complete device failure, where the mutable security context parameters cannot be retrieved, it is expected that a failed joined node is replaced with a new physical device, using a new pledge identifier and a PSK. When such a failure event occurs at the JRC, it is possible that the static information on provisioned pledges, like PSKs and pledge identifiers, can be retrieved through available backups. However, it is likely that the information about joined nodes, their assigned short identifiers and mutable security context parameters is lost. If this is the case, during the process of JRC reinitialization, the network administrator MUST force through out-of-band means all the networks managed by the failed JRC to rejoin, through e.g. the reinitialization of the 6LBR nodes and freshly generated dynamic cryptographic keys and other parameters that have influence on the security properties of the network.

In order to recover from such a failure event, the reinitialized JRC can trigger the renegotiation of the OSCORE security context through the procedure described in Appendix B.2 of [RFC8613]. Aware of the failure event, the reinitialized JRC responds to the first join request of each pledge it is managing with a 4.01 Unauthorized error and a random nonce. The pledge verifies the error response and then initiates the CoJP join exchange using a new OSCORE security context derived from an ID Context consisting of the concatenation of two nonces, one that it received from the JRC and the other that the pledge generates locally. After verifying the join request with the new ID Context and the derived OSCORE security context, the JRC

should consequently take action in mapping the new ID Context with the previously used pledge identifier. How JRC handles this mapping is out of scope of this document.

The described procedure is specified in Appendix B.2 of [RFC8613] and is RECOMMENDED in order to handle the failure events or any other event that may lead to the loss of mutable security context parameters. The length of nonces exchanged using this procedure MUST be at least 8 bytes.

The procedure does require both the pledge and the JRC to have good sources of randomness. While this is typically not an issue at the JRC side, the constrained device hosting the pledge may pose limitations in this regard. If the procedure outlined in Appendix B.2 of [RFC8613] is not supported by the pledge, the network administrator MUST take action in reprovisioning the concerned devices with freshly generated parameters, through out-of-band means.

8.4. CoJP Objects

This section specifies the structure of CoJP CBOR objects that may be carried as the payload of CoJP messages. Some of these objects may be received both as part of the CoJP join exchange when the device operates as a (CoJP) pledge, or the parameter update exchange, when the device operates as a joined (6LBR) node.

8.4.1. Join Request Object

The Join_Request structure is built on a CBOR map object.

The set of parameters that can appear in a Join_Request object is summarized below. The labels can be found in the "CoJP Parameters" registry Section 11.1.

- o role: The identifier of the role that the pledge requests to play in the network once it joins, encoded as an unsigned integer. Possible values are specified in Table 2. This parameter MAY be included. In case the parameter is omitted, the default value of 0, i.e. the role "6TiSCH Node", MUST be assumed.
- o network identifier: The identifier of the network, as discussed in Section 3, encoded as a CBOR byte string. When present in the Join_Request, it hints to the JRC the network that the pledge is requesting to join, enabling the JRC to manage multiple networks. The pledge obtains the value of the network identifier from the received EB frames. This parameter MUST be included in a Join_Request object regardless of the role parameter value.

- o **unsupported configuration:** The identifier of the parameters that are not supported by the implementation, encoded as an `Unsupported_Configuration` object described in Section 8.4.5. This parameter MAY be included. If a (6LBR) pledge previously attempted to join and received a valid Join Response message over OSCORE, but failed to act on its payload (Configuration object), it SHOULD include this parameter to facilitate the recovery and debugging.

Table 1 summarizes the parameters that may appear in a `Join_Request` object.

Name	Label	CBOR Type
role	1	unsigned integer
network identifier	5	byte string
unsupported configuration	8	array

Table 1: Summary of `Join_Request` parameters.

The CDDL fragment that represents the text above for the `Join_Request` follows.

```
Join_Request = {
  ? 1 : uint,           ; role
  5 : bstr,             ; network identifier
  ? 8 : Unsupported_Configuration ; unsupported configuration
}
```

Name	Value	Description	Reference
6TiSCH Node	0	The pledge requests to play the role of a regular 6TiSCH node, i.e. non-6LBR node.	[[this document]]
6LBR	1	The pledge requests to play the role of 6LoWPAN Border Router (6LBR).	[[this document]]

Table 2: Role values.

8.4.2. Configuration Object

The Configuration structure is built on a CBOR map object. The set of parameters that can appear in a Configuration object is summarized below. The labels can be found in "CoJP Parameters" registry Section 11.1.

- o link-layer key set: An array encompassing a set of cryptographic keys and their identifiers that are currently in use in the network, or that are scheduled to be used in the future. The encoding of individual keys is described in Section 8.4.3. The link-layer key set parameter MAY be included in a Configuration object. When present, the link-layer key set parameter MUST contain at least one key. This parameter is also used to implement rekeying in the network. How the keys are installed and used differs for the 6LBR and other (regular) nodes, and this is explained in Section 8.4.3.1 and Section 8.4.3.2.
- o short identifier: a compact identifier assigned to the pledge. The short identifier structure is described in Section 8.4.4. The short identifier parameter MAY be included in a Configuration object.
- o JRC address: the IPv6 address of the JRC, encoded as a byte string, with the length of 16 bytes. If the length of the byte string is different from 16, the parameter MUST be discarded. If the JRC is not co-located with the 6LBR and has a different IPv6 address than the 6LBR, this parameter MUST be included. In the special case where the JRC is co-located with the 6LBR and has the same IPv6 address as the 6LBR, this parameter MAY be included. If the JRC address parameter is not present in the Configuration object, this indicates that the JRC has the same IPv6 address as the 6LBR. The joined node can then discover the IPv6 address of the JRC through network control traffic. See Section 6.
- o blacklist: An array encompassing a list of pledge identifiers that are blacklisted by the JRC, with each pledge identifier encoded as a byte string. The blacklist parameter MAY be included in a Configuration object. When present, the array MUST contain zero or more byte strings encoding pledge identifiers. The joined node MUST silently drop any link-layer frames originating from the pledge identifiers enclosed in the blacklist parameter. When this parameter is received, its value MUST overwrite any previously set values. This parameter allows the JRC to configure the node acting as a JP to filter out traffic from misconfigured or malicious pledges before their traffic is forwarded into the network. If the JRC decides to remove a given pledge identifier from a blacklist, it omits the pledge identifier in the blacklist

parameter value it sends next. Since the blacklist parameter carries the pledge identifiers, privacy considerations apply. See Section 10.

- o join rate: Average data rate (in units of bytes/second) of join traffic forwarded into the network that should not be exceeded when a joined node operates as a JP, encoded as an unsigned integer. The join rate parameter MAY be included in a Configuration object. This parameter allows the JRC to configure different nodes in the network to operate as JP, and act in case of an attack by throttling the rate at which JP forwards unauthenticated traffic into the network. When this parameter is present in a Configuration object, the value MUST be used to set the PROBING_RATE of CoAP at the joined node for communication with the JRC. In case this parameter is set to zero, a joined node MUST silently drop any join traffic coming from unauthenticated pledges. In case this parameter is omitted, the value of positive infinity SHOULD be assumed. Node operating as a JP MAY use another mechanism that is out of scope of this specification to configure PROBING_RATE of CoAP in the absence of a join rate parameter from the Configuration object.

Table 3 summarizes the parameters that may appear in a Configuration object.

Name	Label	CBOR Type
link-layer key set	2	array
short identifier	3	array
JRC address	4	byte string
blacklist	6	array
join rate	7	unsigned integer

Table 3: Summary of Configuration parameters.

The CDDL fragment that represents the text above for the Configuration follows. Structures Link_Layer_Key and Short_Identifier are specified in Section 8.4.3 and Section 8.4.4.

```
Configuration = {  
    ? 2 : [ +Link_Layer_Key ],    ; link-layer key set  
    ? 3 : Short_Identifier,       ; short identifier  
    ? 4 : bstr,                   ; JRC address  
    ? 6 : [ *bstr ],              ; blacklist  
    ? 7 : uint                     ; join rate  
}
```

Name	Label	CBOR type	Description	Reference
role	1	unsigned integer	Identifies the role parameter	[[this document]]
link-layer key set	2	array	Identifies the array carrying one or more link-level cryptographic keys	[[this document]]
short identifier	3	array	Identifies the assigned short identifier	[[this document]]
JRC address	4	byte string	Identifies the IPv6 address of the JRC	[[this document]]
network identifier	5	byte string	Identifies the network identifier parameter	[[this document]]
blacklist	6	array	Identifies the blacklist parameter	[[this document]]
join rate	7	unsigned integer	Identifier the join rate parameter	[[this document]]
unsupported configuration	8	array	Identifies the unsupported configuration parameter	[[this document]]

Table 4: CoJP parameters map labels.

8.4.3. Link-Layer Key

The Link_Layer_Key structure encompasses the parameters needed to configure the link-layer security module: the key identifier; the value of the cryptographic key; the link-layer algorithm identifier

and the security level and the frame types that it should be used with, both for outgoing and incoming security operations; and any additional information that may be needed to configure the key.

For encoding compactness, the `Link_Layer_Key` object is not enclosed in a top-level CBOR object. Rather, it is transported as a sequence of CBOR elements [I-D.ietf-cbor-sequence], some being optional.

The set of parameters that can appear in a `Link_Layer_Key` object is summarized below, in order:

- o `key_id`: The identifier of the key, encoded as a CBOR unsigned integer. This parameter **MUST** be included. If the decoded CBOR unsigned integer value is larger than the maximum link-layer key identifier, the key is considered invalid. In case the key is considered invalid, the key **MUST** be discarded and the implementation **MUST** signal the error as specified in Section 8.3.1.
- o `key_usage`: The identifier of the link-layer algorithm, security level and link-layer frame types that can be used with the key, encoded as an integer. This parameter **MAY** be included. Possible values and the corresponding link-layer settings are specified in IANA "CoJP Key Usage" registry (Section 11.2). In case the parameter is omitted, the default value of 0 (6TiSCH-K1K2-ENC-MIC32) from Table 5 **MUST** be assumed. This default value has been chosen such that it results in byte savings in the most constrained settings but does not imply a recommendation for its general usage.
- o `key_value`: The value of the cryptographic key, encoded as a byte string. This parameter **MUST** be included. If the length of the byte string is different than the corresponding key length for a given algorithm specified by the `key_usage` parameter, the key **MUST** be discarded and the implementation **MUST** signal the error as specified in Section 8.3.1.
- o `key_addinfo`: Additional information needed to configure the link-layer key, encoded as a byte string. This parameter **MAY** be included. The processing of this parameter is dependent on the link-layer technology in use and a particular keying mode.

To be able to decode the keys that are present in the link-layer key set, and to identify individual parameters of a single `Link_Layer_Key` object, the CBOR decoder needs to differentiate between elements based on the CBOR type. For example, a uint that follows a byte string signals to the decoder that a new `Link_Layer_Key` object is being processed.

The CDDL fragment that represents the text above for the Link_Layer_Key follows.

```
Link_Layer_Key = (
    key_id          : uint,
    ? key_usage     : int,
    key_value       : bstr,
    ? key_addinfo   : bstr,
)
```

Name	Value	Algorithm	Description	Reference
6TiSCH-K1K2-ENC-MIC32	0	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, ENC-MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC64	1	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC128	2	IEEE802154-AES-CCM-128	Use MIC-128 for EBs, ENC-MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC32	3	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC64	4	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-	5	IEEE802154-AES-	Use MIC-128	[[this d

K1K2-MIC128			CCM-128	for EBs, DATA and AC KNOWLEDGMEN T.	ocument]]
6TiSCH-K1-MIC32	6		IEEE802154-AES- CCM-128	Use MIC-32 for EBs.	[[this d ocument]]
6TiSCH-K1-MIC64	7		IEEE802154-AES- CCM-128	Use MIC-64 for EBs.	[[this d ocument]]
6TiSCH-K1-MIC128	8		IEEE802154-AES- CCM-128	Use MIC-128 for EBs.	[[this d ocument]]
6TiSCH-K2-MIC32	9		IEEE802154-AES- CCM-128	Use MIC-32 for DATA and ACKNOWL EDGMENT.	[[this d ocument]]
6TiSCH-K2-MIC64	10		IEEE802154-AES- CCM-128	Use MIC-64 for DATA and ACKNOWL EDGMENT.	[[this d ocument]]
6TiSCH-K2-MIC128	11		IEEE802154-AES- CCM-128	Use MIC-128 for DATA and ACKNOWL EDGMENT.	[[this d ocument]]
6TiSCH-K2-ENC- MIC32	12		IEEE802154-AES- CCM-128	Use ENC- MIC-32 for DATA and AC KNOWLEDGMEN T.	[[this d ocument]]
6TiSCH-K2-ENC- MIC64	13		IEEE802154-AES- CCM-128	Use ENC- MIC-64 for DATA and AC KNOWLEDGMEN T.	[[this d ocument]]
6TiSCH-K2-ENC- MIC128	14		IEEE802154-AES- CCM-128	Use ENC- MIC-128 for DATA and AC KNOWLEDGMEN	[[this d ocument]]

			T.	
+-----+	+-----+	+-----+	+-----+	+-----+

Table 5: Key Usage values.

8.4.3.1. Rekeying of (6LoWPAN) Border Routers (6LBR)

When the 6LoWPAN Border Router (6LBR) receives the Configuration object containing a link-layer key set, it MUST immediately install and start using the new keys for all outgoing traffic, and remove any old keys it has installed from the previous key set after a delay of COJP_REKEYING_GUARD_TIME has passed. This mechanism is used by the JRC to force the 6LBR to start sending traffic with the new key. The decision is taken by the JRC when it has determined that the new key has been made available to all (or some overwhelming majority) of nodes. Any node that the JRC has not yet reached at that point is either non-functional or in extended sleep such that it will not be reached. To get the key update, such node needs to go through the join process anew.

8.4.3.2. Rekeying of regular (6LoWPAN) Nodes (6LN)

When a regular 6LN node receives the Configuration object with a link-layer key set, it MUST install the new keys. The 6LN will use both the old and the new keys to decrypt and authenticate any incoming traffic that arrives based upon the key identifier in the packet. It MUST continue to use the old keys for all outgoing traffic until it has detected that the network has switched to the new key set.

The detection of network switch is based upon the receipt of traffic secured with the new keys. Upon reception and successful security processing of a link-layer frame secured with a key from the new key set, a 6LN node MUST then switch to sending outgoing traffic using the keys from the new set for all outgoing traffic. The 6LN node MUST remove any old keys it has installed from the previous key set after a delay of COJP_REKEYING_GUARD_TIME has passed after it starts using the new key set.

Sending of traffic with the new keys signals to other downstream nodes to switch to their new key, and the effect is that there is a ripple of key updates around each 6LBR.

8.4.3.3. Use in IEEE Std 802.15.4

When Link_Layer_Key is used in the context of [IEEE802.15.4], the following considerations apply.

Signaling of different keying modes of [IEEE802.15.4] is done based on the parameter values present in a Link_Layer_Key object. For instance, the value of the key_id parameter in combination with key_addinfo denotes which of the four Key ID modes of [IEEE802.15.4] is used and how.

- o Key ID Mode 0x00 (Implicit, pairwise): key_id parameter MUST be set to 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to the link-layer address(es) of a single peer with whom the key should be used. Depending on the configuration of the network, key_addinfo may carry the peer's long link-layer address (i.e. pledge identifier), short link-layer address, or their concatenation with the long address being encoded first. Which address type(s) is carried is determined from the length of the byte string.
- o Key ID Mode 0x01 (Key Index): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST NOT be present.
- o Key ID Mode 0x02 (4-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 4 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].
- o Key ID Mode 0x03 (8-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 8 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].

In all cases, key_usage parameter determines how a particular key should be used in respect to incoming and outgoing security policies.

For Key ID Modes 0x01 - 0x03, parameter key_id sets the "secKeyIndex" parameter of [IEEE802.15.4] that is signaled in all outgoing frames secured with a given key. The maximum value key_id can have is 254. The value of 255 is reserved in [IEEE802.15.4] and is therefore considered invalid.

Key ID Mode 0x00 (Implicit, pairwise) enables the JRC to act as a trusted third party and assign pairwise keys between nodes in the network. How JRC learns about the network topology is out of scope of this specification, but could be done through 6LBR - JRC signaling for example. Pairwise keys could also be derived through a key agreement protocol executed between the peers directly, where the authentication is based on the symmetric cryptographic material

provided to both peers by the JRC. Such a protocol is out of scope of this specification.

Implementations MUST use different link-layer keys when using different authentication tag (MIC) lengths, as using the same key with different authentication tag lengths might be unsafe. For example, this prohibits the usage of the same key for both MIC-32 and MIC-64 levels. See Annex B.4.3 of [IEEE802.15.4] for more information.

8.4.4. Short Identifier

The `Short_Identifier` object represents an identifier assigned to the pledge. It is encoded as a CBOR array object, containing, in order:

- o `identifier`: The short identifier assigned to the pledge, encoded as a byte string. This parameter MUST be included. The identifier MUST be unique in the set of all identifiers assigned in a network that is managed by a JRC. In case the identifier is invalid, the decoder MUST silently ignore the `Short_Identifier` object.
- o `lease_time`: The validity of the identifier in hours after the reception of the CBOR object, encoded as a CBOR unsigned integer. This parameter MAY be included. The node MUST stop using the assigned short identifier after the expiry of the `lease_time` interval. It is up to the JRC to renew the lease before the expiry of the previous interval. The JRC updates the lease by executing the Parameter Update exchange with the node and including the `Short_Identifier` in the Configuration object, as described in Section 8.2. In case the lease expires, the node SHOULD initiate a new join exchange, as described in Section 8.1. In case this parameter is omitted, the value of positive infinity MUST be assumed, meaning that the identifier is valid for as long as the node participates in the network.

The CDDL fragment that represents the text above for the `Short_Identifier` follows.

```
Short_Identifier = [  
    identifier      : bstr,  
    ? lease_time    : uint  
]
```

8.4.4.1. Use in IEEE Std 802.15.4

When `Short_Identifier` is used in the context of [IEEE802.15.4], the following considerations apply.

The identifier **MUST** be used to set the short address of IEEE Std 802.15.4 module. When operating in TSCH mode, the identifier **MUST** be unique in the set of all identifiers assigned in multiple networks that share link-layer key(s). If the length of the byte string corresponding to the identifier parameter is different than 2, the identifier is considered invalid. The values 0xfffe and 0xffff are reserved by [IEEE802.15.4] and their use is considered invalid.

The security properties offered by the [IEEE802.15.4] link-layer in TSCH mode are conditioned on the uniqueness requirement of the short identifier (i.e. short address). The short address is one of the inputs in the construction of the nonce, which is used to protect link-layer frames. If a misconfiguration occurs, and the same short address is assigned twice under the same link-layer key, the loss of security properties is imminent. For this reason, practices where the pledge generates the short identifier locally are not safe and are likely to result in the loss of link-layer security properties.

The JRC **MUST** ensure that at any given time there are never two same short identifiers being used under the same link-layer key. If the `lease_time` parameter of a given `Short_Identifier` object is set to positive infinity, care needs to be taken that the corresponding identifier is not assigned to another node until the JRC is certain that it is no longer in use, potentially through out-of-band signaling. If the `lease_time` parameter expires for any reason, the JRC should take into consideration potential ongoing transmissions by the joined node, which may be hanging in the queues, before assigning the same identifier to another node.

Care needs to be taken on how the pledge (joined node) configures the expiration of the lease. Since units of the `lease_time` parameter are in hours after the reception of the CBOR object, the pledge needs to convert the received time to the corresponding absolute slot number in the network. The joined node (pledge) **MUST** only use the absolute slot number as the appropriate reference of time to determine whether the assigned short identifier is still valid.

8.4.5. Unsupported Configuration Object

The `Unsupported_Configuration` object is encoded as a CBOR array, containing at least one `Unsupported_Parameter` object. Each `Unsupported_Parameter` object is a sequence of CBOR elements without an enclosing top-level CBOR object for compactness. The set of

parameters that appear in an `Unsupported_Parameter` object is summarized below, in order:

- o `code`: Indicates the capability of acting on the parameter signaled by `parameter_label`, encoded as an integer. This parameter **MUST** be included. Possible values of this parameter are specified in the IANA "CoJP Unsupported Configuration Code Registry" (Section 11.3).
- o `parameter_label`: Indicates the parameter. This parameter **MUST** be included. Possible values of this parameter are specified in the label column of the IANA "CoJP Parameters" registry (Section 11.1).
- o `parameter_addinfo`: Additional information about the parameter that cannot be acted upon. This parameter **MUST** be included. In case the code is set to "Unsupported", `parameter_addinfo` gives additional information to the JRC. If the parameter indicated by `parameter_label` cannot be acted upon regardless of its value, `parameter_addinfo` **MUST** be set to null, signaling to the JRC that it **SHOULD NOT** attempt to configure the parameter again. If the pledge can act on the parameter, but cannot configure the setting indicated by the parameter value, the pledge can hint this to the JRC. In this case, `parameter_addinfo` **MUST** be set to the value of the parameter that cannot be acted upon following the normative parameter structure specified in this document. For example, it is possible to include the link-layer key set object, signaling a subset of keys that cannot be acted upon, or the entire key set that was received. In that case, the value of the `parameter_addinfo` follows the link-layer key set structure defined in Section 8.4.2. In case the code is set to "Malformed", `parameter_addinfo` **MUST** be set to null, signaling to the JRC that it **SHOULD NOT** attempt to configure the parameter again.

The CDDL fragment that represents the text above for `Unsupported_Configuration` and `Unsupported_Parameter` objects follows.

```
Unsupported_Configuration = [  
    + parameter           : Unsupported_Parameter  
]  
  
Unsupported_Parameter = (  
    code                  : int,  
    parameter_label       : int,  
    parameter_addinfo     : nil / any  
)
```

Name	Value	Description	Reference
Unsupported	0	The indicated setting is not supported by the networking stack implementation.	[[this document]]
Malformed	1	The indicated parameter value is malformed.	[[this document]]

Table 6: Unsupported Configuration code values.

8.5. Recommended Settings

This section gives RECOMMENDED values of CoJP settings.

Name	Default Value
COJP_MAX_JOIN_ATTEMPTS	4
COJP_REKEYING_GUARD_TIME	12 seconds

Recommended CoJP settings.

The COJP_REKEYING_GUARD_TIME value SHOULD take into account possible retransmissions at the link layer due to imperfect wireless links.

9. Security Considerations

Since this document uses the pledge identifier to set the ID Context parameter of OSCORE, an important security requirement is that the pledge identifier is unique in the set of all pledge identifiers managed by a JRC. The uniqueness of the pledge identifier ensures unique (key, nonce) pairs for AEAD algorithm used by OSCORE. It also allows the JRC to retrieve the correct security context, upon the reception of a Join Request message. The management of pledge identifiers is simplified if the globally unique EUI-64 is used, but this comes with privacy risks, as discussed in Section 10.

This document further mandates that the (6LBR) pledge and the JRC are provisioned with unique PSKs. While the process of provisioning PSKs to all pledges can result in a substantial operational overhead, it is vital to do so for the security properties of the network. The PSK is used to set the OSCORE Master Secret during security context derivation. This derivation process results in OSCORE keys that are

important for mutual authentication of the (6LBR) pledge and the JRC. The resulting security context shared between the pledge (joined node) and the JRC is used for the purpose of joining and is long-lived in that it can be used throughout the lifetime of a joined node for parameter update exchanges. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible.

Note that while OSCORE provides replay protection, it does not provide an indication of freshness in the presence of an attacker that can drop/reorder traffic. Since the join request contains no randomness, and the sequence number is predictable, the JRC could in principle anticipate a join request from a particular pledge and pre-calculate the response. In such a scenario, the JRC does not have to be alive at the time when the request is received. This could be relevant in case the JRC was temporarily compromised and control subsequently regained by the legitimate owner.

It is of utmost importance to avoid unsafe practices when generating and provisioning PSKs. The use of a single PSK shared among a group of devices is a common pitfall that results in poor security. In this case, the compromise of a single device is likely to lead to a compromise of the entire batch, with the attacker having the ability to impersonate a legitimate device and join the network, generate bogus data and disturb the network operation. Additionally, some vendors use methods such as scrambling or hashing of device serial numbers or their EUI-64 to generate "unique" PSKs. Without any secret information involved, the effort that the attacker needs to invest into breaking these unsafe derivation methods is quite low, resulting in the possible impersonation of any device from the batch, without even needing to compromise a single device. The use of cryptographically secure random number generators to generate the PSK is RECOMMENDED, see [NIST800-90A] for different mechanisms using deterministic methods.

The JP forwards the unauthenticated join traffic into the network. A data cap on the JP prevents it from forwarding more traffic than the network can handle and enables throttling in case of an attack. Note that this traffic can only be directed at the JRC so that the JRC needs to be prepared to handle such unsanitized inputs. The data cap can be configured by the JRC by including a join rate parameter in the Join Response and it is implemented through the CoAP's PROBING_RATE setting. The use of a data cap at a JP forces attackers to use more than one JP if they wish to overwhelm the network. Marking the join traffic packets with a non-zero DSCP allows the network to carry the traffic if it has capacity, but encourages the network to drop the extra traffic rather than add bandwidth due to that traffic.

The shared nature of the "minimal" cell used for the join traffic makes the network prone to a DoS attack by congesting the JP with bogus traffic. Such an attacker is limited by its maximum transmit power. The redundancy in the number of deployed JPs alleviates the issue and also gives the pledge a possibility to use the best available link for joining. How a network node decides to become a JP is out of scope of this specification.

At the beginning of the join process, the pledge has no means of verifying the content in the EB, and has to accept it at "face value". In case the pledge tries to join an attacker's network, the Join Response message will either fail the security check or time out. The pledge may implement a temporary blacklist in order to filter out undesired EBs and try to join using the next seemingly valid EB. This blacklist alleviates the issue, but is effectively limited by the node's available memory. Note that this temporary blacklist is different from the one communicated as part of the CoJP Configuration object as it helps pledge fight a DoS attack. The bogus beacons prolong the join time of the pledge, and so the time spent in "minimal" [RFC8180] duty cycle mode. The blacklist communicated as part of the CoJP Configuration object helps JP fight a DoS attack by a malicious pledge.

During the network lifetime, the JRC may at any time initiate a Parameter Update exchange with a joined node. The Parameter Update message uses the same OSCORE security context as is used for the join exchange, except that the server/client roles are interchanged. As a consequence, each Parameter Update message carries the well-known OSCORE Sender ID of the JRC. A passive attacker may use the OSCORE Sender ID to identify the Parameter Update traffic in case the link-layer protection does not provide confidentiality. A countermeasure against such traffic analysis attack is to use encryption at the link-layer. Note that the join traffic does not undergo link-layer protection at the first hop, as the pledge is not yet in possession of cryptographic keys. Similarly, enhanced beacon traffic in the network is not encrypted. This makes it easy for a passive attacker to identify these types of traffic.

10. Privacy Considerations

The join solution specified in this document relies on the uniqueness of the pledge identifier in the set of all pledge identifiers managed by a JRC. This identifier is transferred in clear as an OSCORE kid context. The use of the globally unique EUI-64 as pledge identifier simplifies the management but comes with certain privacy risks. The implications are thoroughly discussed in [RFC7721] and comprise correlation of activities over time, location tracking, address scanning and device-specific vulnerability exploitation. Since the

join process occurs rarely compared to the network lifetime, long-term threats that arise from using EUI-64 as the pledge identifier are minimal. However, the use of EUI-64 after the join process completes, in the form of a layer-2 or layer-3 address, extends the aforementioned privacy threats to long term.

As an optional mitigation technique, the Join Response message may contain a short address which is assigned by the JRC to the (6LBR) pledge. The assigned short address SHOULD be uncorrelated with the long-term pledge identifier. The short address is encrypted in the response. Once the join process completes, the new node may use the short addresses for all further layer-2 (and layer-3) operations. This reduces the privacy threats as the short layer-2 address (visible even when the network is encrypted) does not disclose the manufacturer, as is the case of EUI-64. However, an eavesdropper with access to the radio medium during the join process may be able to correlate the assigned short address with the extended address based on timing information with a non-negligible probability. This probability decreases with an increasing number of pledges joining concurrently.

11. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "[[this document]]" with the RFC number of this specification.

This document allocates a well-known name under the .arpa name space according to the rules given in [RFC3172]. The name "6tisch.arpa" is requested. No subdomains are expected, and addition of any such subdomains requires the publication of an IETF standards-track RFC. No A, AAAA or PTR record is requested.

11.1. CoJP Parameters Registry

This section defines a sub-registry within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Parameters Registry".

The columns of the registry are:

Name: This is a descriptive name that enables an easier reference to the item. It is not used in the encoding.

Label: The value to be used to identify this parameter. The label is an integer.

CBOR type: This field contains the CBOR type for the field.

Description: This field contains a brief description for the field.

Reference: This field contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 4.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

11.2. CoJP Key Usage Registry

This section defines a sub-registry within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Key Usage Registry".

The columns of this registry are:

Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

Value: This is the value used to identify the key usage setting. These values MUST be unique. The value is an integer.

Algorithm: This is a descriptive name of the link-layer algorithm in use and uniquely determines the key length. The name is not used in the encoding.

Description: This field contains a description of the key usage setting. The field should describe in enough detail how the key is to be used with different frame types, specific for the link-layer technology in question.

Reference: This contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 5.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are

designated as Expert Review. Integer values less than -65536 are marked as Private Use.

11.3. CoJP Unsupported Configuration Code Registry

This section defines a sub-registry within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Unsupported Configuration Code Registry".

The columns of this registry are:

Name: This is a descriptive name that enables easier reference to the item. The name **MUST** be unique. It is not used in the encoding.

Value: This is the value used to identify the diagnostic code. These values **MUST** be unique. The value is an integer.

Description: This is a descriptive human-readable name. The description **MUST** be unique. It is not used in the encoding.

Reference: This contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 6.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

12. Acknowledgments

The work on this document has been partially supported by the European Union's H2020 Programme for research, technological development and demonstration under grant agreements: No 644852, project ARMOUR; No 687884, project F-Interop and open-call project SPOTS; No 732638, project Fed4FIRE+ and open-call project SODA.

The following individuals provided input to this document (in alphabetic order): Christian Amsuss, Tengfei Chang, Klaus Hartke, Tero Kivinen, Jim Schaad, Goeran Selander, Yasuyuki Tanaka, Pascal Thubert, William Vignat, Xavier Vilajosana, Thomas Watteyne.

13. References

13.1. Normative References

- [I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-28 (work in progress), October 2019.
- [I-D.ietf-core-stateless]
Hartke, K., "Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)", draft-ietf-core-stateless-03 (work in progress), October 2019.
- [IEEE802.15.4]
IEEE standard for Information Technology, ., "IEEE Std 802.15.4 Standard for Low-Rate Wireless Networks", n.d..
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <<https://www.rfc-editor.org/info/rfc2597>>.
- [RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <<https://www.rfc-editor.org/info/rfc3172>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [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>>.

- [RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <<https://www.rfc-editor.org/info/rfc7320>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<https://www.rfc-editor.org/info/rfc7554>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <<https://www.rfc-editor.org/info/rfc8180>>.
- [RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <<https://www.rfc-editor.org/info/rfc8505>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.

13.2. Informative References

- [I-D.ietf-6tisch-msf]
Chang, T., Vucinic, M., Vilajosana, X., Duquennoy, S., and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)", draft-ietf-6tisch-msf-08 (work in progress), November 2019.
- [I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-15 (work in progress), July 2017.
- [I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", draft-ietf-cbor-cddl-08 (work in progress), March 2019.
- [I-D.ietf-cbor-sequence]
Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", draft-ietf-cbor-sequence-02 (work in progress), September 2019.
- [NIST800-90A]
NIST Special Publication 800-90A, Revision 1, ., Barker, E., and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", 2015.
- [RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <<https://www.rfc-editor.org/info/rfc4231>>.
- [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>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<https://www.rfc-editor.org/info/rfc5785>>.

- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [RFC8480] Wang, Q., Ed., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)", RFC 8480, DOI 10.17487/RFC8480, November 2018, <<https://www.rfc-editor.org/info/rfc8480>>.

Appendix A. Example

Figure 3 illustrates a successful join protocol exchange. The pledge instantiates the OSCORE context and derives the OSCORE keys and nonces from the PSK. It uses the instantiated context to protect the Join Request addressed with a Proxy-Scheme option, the well-known host name of the JRC in the Uri-Host option, and its EUI-64 as pledge identifier and OSCORE kid context. Triggered by the presence of a Proxy-Scheme option, the JP forwards the request to the JRC and sets the CoAP token to the internally needed state. The JP has learned the IPv6 address of the JRC when it acted as a pledge and joined the network. Once the JRC receives the request, it looks up the correct context based on the kid context parameter. The OSCORE data authenticity verification ensures that the request has not been modified in transit. In addition, replay protection is ensured through persistent handling of mutable context parameters.

Once the JP receives the Join Response, it authenticates the state within the CoAP token before deciding where to forward. The JP sets its internal state to that found in the token, and forwards the Join Response to the correct pledge. Note that the JP does not possess

the key to decrypt the CoJP object (configuration) present in the payload. The Join Response is matched to the Join Request and verified for replay protection at the pledge using OSCORE processing rules. In this example, the Join Response does not contain the IPv6 address of the JRC, the pledge hence understands the JRC is co-located with the 6LBR.

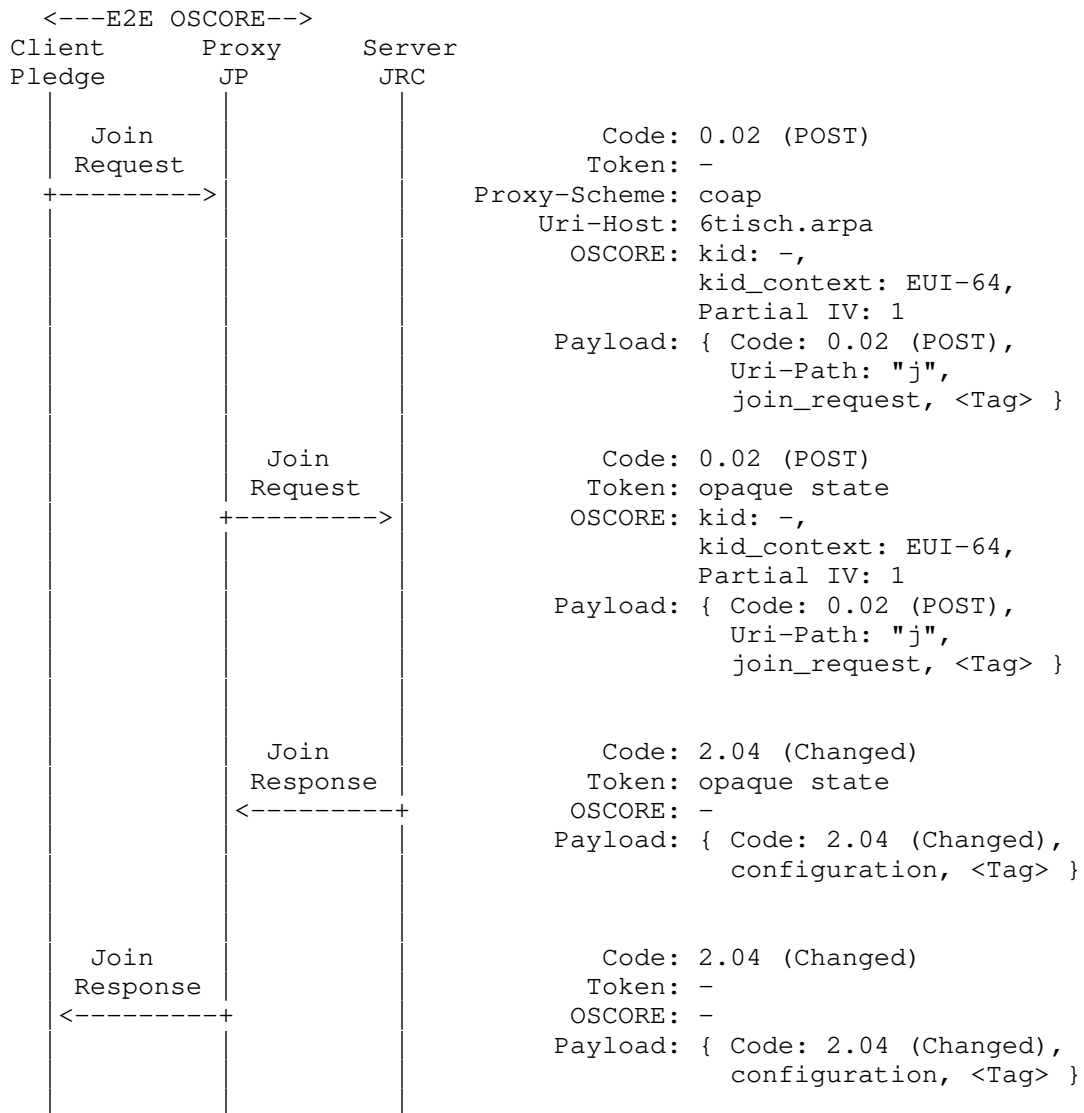


Figure 3: Example of a successful join protocol exchange. { ... } denotes authenticated encryption, <Tag> denotes the authentication tag.

Where the join_request object is:

```
join_request:
{
    5 : h'cafe' / PAN ID of the network pledge is attempting to join /
}
```

Since the role parameter is not present, the default role of "6TiSCH Node" is implied.

The join_request object encodes to h'a10542cafe' with a size of 5 bytes.

And the configuration object is:

```
configuration:
{
    2 : [          / link-layer key set /
        1,          / key_id /
        h'e6bf4287c2d7618d6a9687445ffd33e6' / key_value /
    ],
    3 : [          / short identifier /
        h'af93'     / assigned short address /
    ]
}
```

Since the key_usage parameter is not present in the link-layer key set object, the default value of "6TiSCH-K1K2-ENC-MIC32" is implied. Since key_addinfo parameter is not present and key_id is different than 0, Key ID Mode 0x01 (Key Index) is implied. Similarly, since the lease_time parameter is not present in the short identifier object, the default value of positive infinity is implied.

The configuration object encodes to

h'a202820150e6bf4287c2d7618d6a9687445ffd33e6038142af93' with a size of 26 bytes.

Appendix B. Lightweight Implementation Option

In environments where optimizing the implementation footprint is important, it is possible to implement this specification without having the implementations of HKDF [RFC5869] and SHA [RFC4231] on constrained devices. HKDF and SHA are used during the OSCORE security context derivation phase. This derivation can also be done by the JRC or a provisioning device, on behalf of the (6LBR) pledge during the provisioning phase. In that case, the derived OSCORE security context parameters are written directly into the (6LBR) pledge, without requiring the PSK be provisioned to the (6LBR) pledge.

The use of HKDF to derive OSCORE security context parameters ensures that the resulting OSCORE keys have good security properties, and are unique as long as the input for different pledges varies. This specification ensures the uniqueness by mandating unique pledge identifiers and a unique PSK for each (6LBR) pledge. From the AEAD nonce reuse viewpoint, having a unique pledge identifier is a sufficient condition. However, as discussed in Section 9, the use of a single PSK shared among many devices is a common security pitfall. The compromise of this shared PSK on a single device would lead to the compromise of the entire batch. When using the implementation/deployment scheme outlined above, the PSK does not need to be written to individual pledges. As a consequence, even if a shared PSK is used, the scheme offers a comparable level of security as in the scenario where each pledge is provisioned with a unique PSK. In this case, there is still a latent risk of the shared PSK being compromised from the provisioning device, which would compromise all devices in the batch.

Authors' Addresses

Malisa Vucinic (editor)
Inria
2 Rue Simone Iff
Paris 75012
France

Email: malisa.vucinic@inria.fr

Jonathan Simon
Analog Devices
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: jonathan.simon@analog.com

Kris Pister
University of California Berkeley
512 Cory Hall
Berkeley, CA 94720
USA

Email: pister@eecs.berkeley.edu

Michael Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z5V7
Canada

Email: mcr+ietf@sandelman.ca

6lo
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

Lijo Thomas
C-DAC
P. Akshay
Indian Institute of Science
Satish Anamalamudi
Huaiyin Institute of Technology
S.V.R.Anand
Malati Hegde
Indian Institute of Science
C. Perkins
Futurewei
July 3, 2017

Packet Delivery Deadline time in 6LoWPAN Routing Header
draft-lijo-6lo-expiration-time-04

Abstract

This document specifies a new type for the 6LoWPAN routing header containing the delivery deadline time for data packets. The deadline time enables forwarding and scheduling decisions for time critical IoT M2M applications that need deterministic delay guarantees over constrained networks and operate within time-synchronized networks.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 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	2
2. Terminology	3
3. 6LoRHE Generic Format	3
4. Deadline-6LoRHE	3
5. Deadline-6LoRHE Format	4
6. Deadline-6LoRHE in Three Network Scenarios	6
6.1. Scenario 1: Endpoints in the same DODAG (N1) in non-storing mode.	6
6.2. Scenario 2: Endpoints in Networks with Dissimilar L2 Technologies.	7
6.3. Scenario 3: Packet transmission across different DODAGs (N1 to N2).	8
7. IANA Considerations	9
8. Security Considerations	10
9. Acknowledgements	10
10. References	10
10.1. Normative References	10
10.2. Informative References	11
Authors' Addresses	12

1. Introduction

Low Power and Lossy Networks (LLNs) are likely to be deployed for real time industrial applications requiring end-to-end delay guarantees [I-D.grossman-detnet-use-cases]. A Deterministic Network ("detnet") typically requires some data packets to reach their receivers within strict time bounds. Intermediate nodes use the deadline information to make appropriate packet forwarding and scheduling decisions to meet the time bounds.

The draft [I-D.ietf-roll-routing-dispatch] specifies the 6LoWPAN Routing Header (6LoRH), compression schemes for RPL routing (source routing) operation [RFC6554], header compression of RPL Packet Information [RFC6553], and IP-in-IP encapsulation. This document specifies a new Deadline-6LoRHE type for the 6LoWPAN Dispatch Page 1, so that the deadline time of data packets can be included within the

6LoWPAN routing header. This document also specifies handling of the deadline time when packets traverse through time-synchronized networks operating in different timezones or distinct reference clocks.

2. Terminology

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

This document uses terminology consistent with the terminology used in [RFC6550] and [I-D.ietf-6tisch-terminology]. Also, in this document, the terms "expiration time", "delivery deadline time", and "deadline" are used interchangeably with the same meaning.

3. 6LoRHE Generic Format

Note: this section is not normative. It is included for convenience, and may be deleted in a later revision of this document. The generic header format of the 6LoRHE is specified in [I-D.ietf-roll-routing-dispatch]. Figure 1 illustrates the 6LoRHE generic format.

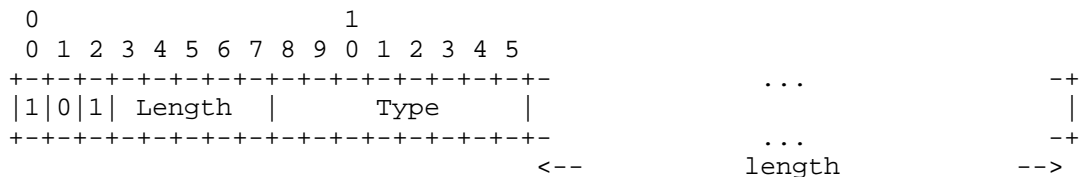


Figure 1: 6LoRHE format

- o Length: Length of the 6LoRHE expressed in bytes, excluding the first 2 bytes. This enables a node to skip a 6LoRHE if the Type is not recognized/supported.
- o Type: Type of the 6LoRHE.
- o length: variable

4. Deadline-6LoRHE

The Deadline-6LoRHE (see Figure 2) is an elective 6LoRH (i.e., a 6LoRHE) that provides the deadline time (DT) for an IPv6 datagram in a compressed form. Along with the deadline, the header can include the packet Origination Time (OT), to enable a close estimate of the

total delay incurred by a packet. The OT field is initialized by the sender using the current time at the outgoing network interface through which the packet is forwarded.

The deadline field contains the value of the delivery deadline time for the packet. The packet SHOULD be delivered to the Receiver before this time.

$$\text{packet_deadline_time} = \text{packet_origination_time} + \text{max_delay}$$

All nodes within the network SHOULD process the Deadline-6LoRHE in order to support delay-sensitive deterministic applications. The packet deadline time (DT) and origination time (OT) are represented in time units determined by a scaling parameter in the routing header. One of the time units is the Network ASN (Absolute Slot Number) which can be used in case of a time slotted synchronized network, for instance a 6TiSCH network, where global time is maintained in the units of slot lengths of a certain resolution.

The delay experienced by packets in the network is a useful metric for network diagnostics and performance monitoring. Whenever the packets crosses into a network using a different reference clock, the Origination Time field is updated to represent the same Origination Time as expressed using the reference clock of the outgoing interface into the new network. This is the same as the current time when the packet is transmitted into the new network, minus the delay already experienced by the packet, say 't'. In effect, to the newly entered network, the packet will appear to have originated 't' time units earlier with respect to the reference clock of the new network.

$$\text{Origination Time in new network} = \text{current_time_in_new_network} - \text{delay_already_experienced_in_previous_network(s)}$$

5. Deadline-6LoRHE Format

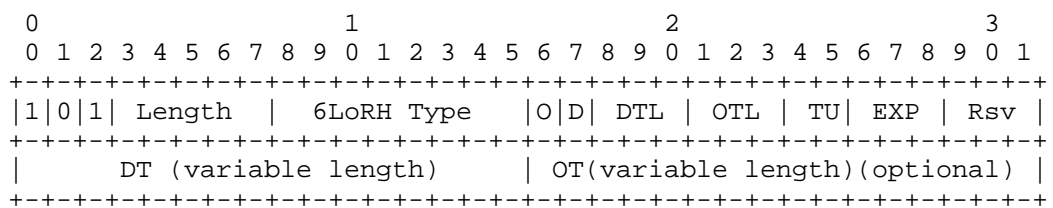


Figure 2: Deadline-6LoRHE format

Length (5 bits): Length represents the total length of the Deadline-6LoRHE type measured in octets.

6LoRH Type: TBD

O flag (1bit): Indicates the presence of Origination Time field. '1' means the OT field is present, and '0' means it is absent.

D flag (1 bit): The 'D' flag, set by the Sender, indicates the action to be taken when a 6LR detects that the deadline time has elapsed. If 'D' bit is 1, then the 6LR SHOULD drop the packet if the deadline time is elapsed. If 'D' bit is 0, then the 6LR MAY ignore the deadline time and forward the packet.

DTL (3 bits): Length of DT field.

OTL (3 bits) : Length of OT field.

For example, DTL = 000 means the deadline time in the 6LoRHE is 1 octet (8 bits) long. Similarly, OTL = 111 means the origination time is 8 octets (64 bits) long.

TU (2 bits) : Indicates the time units for DT and OT fields

00 : Time represented in microseconds
01 : Time represented in seconds
10 : Network ASN
11 : Reserved

EXP (3 bits) : Multiplication factor expressed as exponent of 10.

The value of the DT field is multiplied by 10 to this power, to get the actual deadline time in the units represented by TU. The default value of EXP is 000, so that the DT field is unaffected.

Rsv (3 bits) : Reserved

DT Value (8..64-bit) : Deadline Time value

OT Value (8..64-bit) : Origination Time value

Whenever a sender initiates the IP datagram, it includes the Deadline-6LoRHE along with other 6LoRH information.

Example: Consider a 6TiSCH network with time-slot length of 10ms. Let the current ASN when the packet is originated be 54400, and the maximum allowable delay (max_delay) for the packet delivery is 1 second from the packet origination, then:

$$\text{deadline_time} = \text{packet_origination_time} + \text{max_delay}$$

= 55400 + 100 (in Network ASNs)
 = 55500(Network ASNs)

Deadline-6LoRHE encoding with 'O' flag set to 1 :

DTL = 001, OTL = 001, TU = '10', EXP = 2, DT = 0x22B, OT = 0x22A

6. Deadline-6LoRHE in Three Network Scenarios

In this section, Deadline-6LoRHE operation is described for 3 network scenarios. Figure 3 depicts a constrained time-synchronized LLN that has two subnets N1 and N2, connected through LBRs [I-D.ietf-6lo-backbone-router] with different reference clock times T1 and T2.

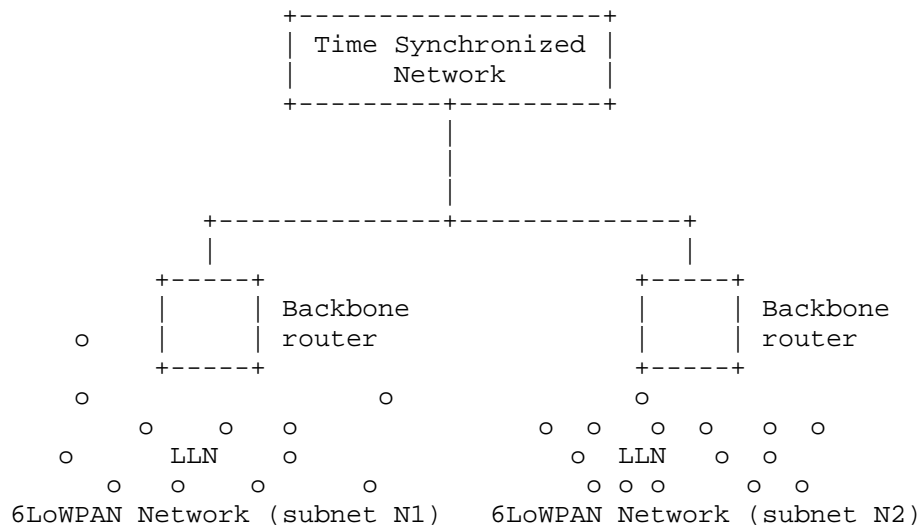


Figure 3: Intra-network Timezone Scenario

6.1. Scenario 1: Endpoints in the same DODAG (N1) in non-storing mode.

In scenario 1, shown in Figure 4, the Sender 'S' has an IP datagram to be routed to a Receiver 'R' within the same DODAG. For the route segment from Sender to 6LBR, the Sender includes a Deadline-6LoRHE by encoding the deadline time contained in the inband-OAM header extension. Then 6LR begins hop-by-hop operation to forward the packet towards the 6LBR. Once 6LBR receives the IP datagram, it generates a IPv6-in-IPv6 encapsulated packet when sending the packet downwards to the Receiver [I-D.ietf-roll-useofrplinfo]. The 6LBR copies the Deadline-6LoRHE from the Sender originated IP header to

the outer IP header. The Deadline-6LoRHE contained in the inner IP header is elided.

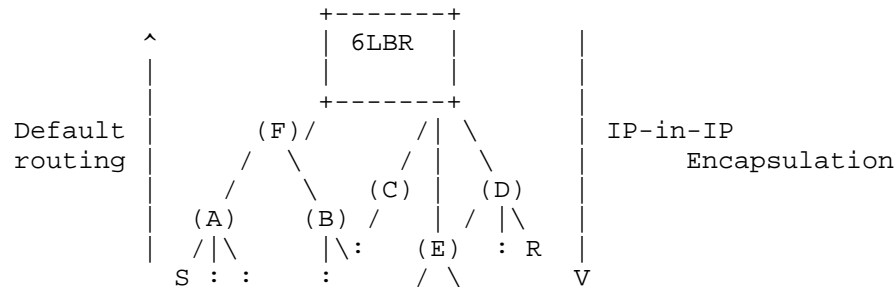


Figure 4: End points within same DODAG(subnet N1)

At the tunnel endpoint of IPv6-in-IPv6 encapsulation, the Deadline-6LoRHE is copied back from the outer header to inner header, and the inner IP packet is delivered to 'R'.

6.2. Scenario 2: Endpoints in Networks with Dissimilar L2 Technologies.

In scenario 2, shown in Figure 5, the Sender 'S' (belonging to DODAG 1) has IP datagram to be routed to a Receiver 'R' over a time-synchronized IPv6 network. For the route segment from 'S' to 6LBR, 'S' includes a Deadline-6LoRHE. Subsequently, 6LR will perform hop-by-hop operation to forward the packet towards the 6LBR. Once the IP datagram reaches 6LBR of DODAG1, it encodes the deadline time (and, if available, the origination time) into the In-band OAM header extension, [I-D.brockners-inband-oam-data] and passes the datagram to the IPv6 layer for further routing.

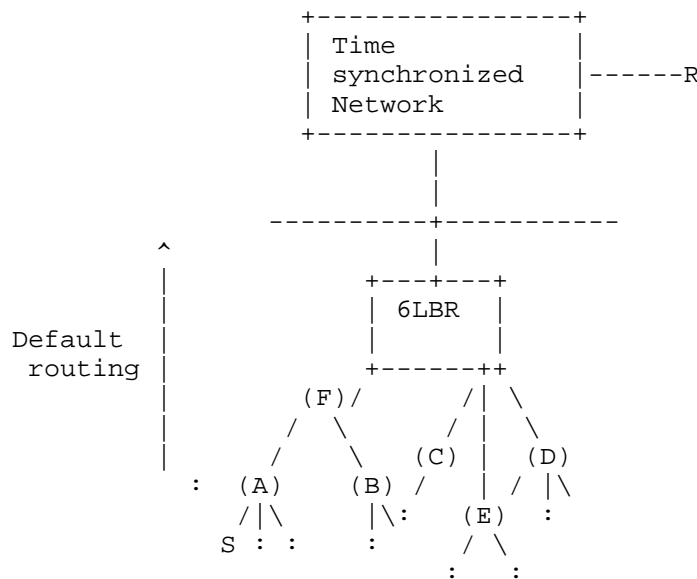


Figure 5: Packet transmission in Dissimilar L2 Technologies or Internet

The IP datagram is routed to another time synchronized deterministic network following its own distinct reference clock, so the deadline time in In-band OAM has to be updated according to the measurement of the current time in the new network.

6.3. Scenario 3: Packet transmission across different DODAGs (N1 to N2).

Consider the scenario depicted in Figure 6, in which the Sender 'S' (belonging to DODAG 1) has an IP datagram to be sent to Receiver 'R' belonging to another DODAG (DODAG 2). The operation of this scenario can be decomposed into combination of case 1 and case 2 scenarios. For the route segment from 'S' to 6LBR, 'S' includes the Deadline-6LoRHE. Subsequently, each 6LR will perform hop-by-hop operation to forward the packet towards the 6LBR. Once the IP datagram reaches 6LBR1 of DODAG1, it applies the same rule as described in Case 2 while routing the packet to LBR2 over a (likely) time synchronized wired backhaul. The wired side of LBR2 can be mapped to receiver of Case 2. Once the packet reaches LBR2, it updates the Deadline-6LoRHE by adding the current time of DODAG2. Further, it generates an IPv6-in-IPv6 encapsulated packet when sending the packet downstream to the Receiver [I-D.ietf-roll-useofrplinfo].

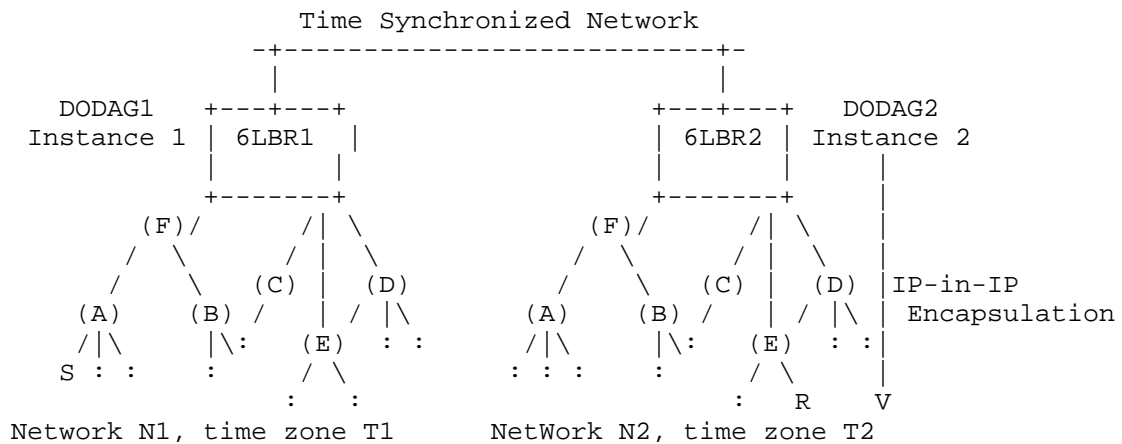


Figure 6: Packet transmission in different DODAGs(N1 to N2)

Consider an example of a 6TiSCH network in which S in DODAG1 generates the packet at ASN 20000 to R in DODAG2. Let the maximum allowable delay be 1 second. The time-slot length in DODAG1 and DODAG2 is assumed to be 10ms. Once the deadline time is encoded in Deadline-6LoRHE, the packet is forwarded to LBR of DODAG1. Suppose the packet reaches LBR of DODAG1 at ASN 20050.

```
current_time = ASN at LBR * slot_length_value
```

```
remaining_time = deadline_time - current_time
= ((packet_origination_time + max_delay) - current time)
= (20000 + 100) - 20050
= 50 (in Network ASNs)
= 50 * 10^3 milliseconds.
```

The remaining time is encoded in In-Band OAM (see Case 2) and forwarded to LBR2 over a different L2-interface, typically wired. Once the packet reaches LBR2, the deadline time in Deadline-6LoRHE is adjusted by adding or subtracting the difference between the reference clocks of the two networks, before forwarding the packet to its connected 6TiSCH network.

7. IANA Considerations

This document defines a new 6LoWPAN Timestamp Header Type, and assigns a value (TBD) from the 6LoWPAN Dispatch Page1 number space.

6LoRH Type	Value
Deadline-6LoRHE	TBD

Figure 7: Deadline-6LoRHE type

8. Security Considerations

The security considerations of [RFC4944], [RFC6282] and [RFC6553] apply. Using a compressed format as opposed to the full in-line format is logically equivalent and does not create an opening for a new threat when compared to [RFC6550], [RFC6553] and [RFC6554].

9. Acknowledgements

The authors thank Pascal Thubert for suggesting the idea and encouraging the work. Thanks to Shwetha Bhandari's suggestions which were instrumental in extending the timing information to heterogeneous networks. The authors acknowledge the 6TiSCH WG members for their inputs on the mailing list. Special thanks to Jerry Daniel, Shalu Rajendran, Seema Kumar, Avinash Mohan and Anita Varghese for their support and valuable feedback.

10. References

10.1. Normative References

- [I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang,
"Terminology in IPv6 over the TSCH mode of IEEE
802.15.4e", draft-ietf-6tisch-terminology-09 (work in
progress), June 2017.
- [I-D.ietf-roll-routing-dispatch]
Thubert, P., Bormann, C., Toutain, L., and R. Cragie,
"6LoWPAN Routing Header", draft-ietf-roll-routing-
dispatch-05 (work in progress), October 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler,
"Transmission of IPv6 Packets over IEEE 802.15.4
Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007,
<<http://www.rfc-editor.org/info/rfc4944>>.

- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<http://www.rfc-editor.org/info/rfc6282>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<http://www.rfc-editor.org/info/rfc6550>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<http://www.rfc-editor.org/info/rfc6553>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<http://www.rfc-editor.org/info/rfc6554>>.

10.2. Informative References

- [I-D.brockners-inband-oam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., <>, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-brockners-inband-oam-data-05 (work in progress), May 2017.
- [I-D.grossman-detnet-use-cases]
Grossman, E., Gunther, C., Thubert, P., Wetterwald, P., Raymond, J., Korhonen, J., Kaneko, Y., Das, S., and Y. Zha, "Deterministic Networking Use Cases", draft-grossman-detnet-use-cases-01 (work in progress), November 2015.
- [I-D.ietf-6lo-backbone-router]
Thubert, P., "IPv6 Backbone Router", draft-ietf-6lo-backbone-router-03 (work in progress), January 2017.
- [I-D.ietf-roll-useofrplinfo]
Robles, I., Richardson, M., and P. Thubert, "When to use RFC 6553, 6554 and IPv6-in-IPv6", draft-ietf-roll-useofrplinfo-15 (work in progress), June 2017.

[I-D.vilajosana-6tisch-minimal]

Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-vilajosana-6tisch-minimal-00 (work in progress), October 2013.

Authors' Addresses

Lijo Thomas
C-DAC
Trivandrum 695033
India

Email: lijo@cdac.in

P.M. Akshay
Indian Institute of Science
Bangalore 560012
India

Email: akshaypm@ece.iisc.ernet.in

Satish Anamalamudi
Huaiyin Institute of Technology
No.89 North Beijing Road, Qinghe District
Huaian
China

Email: satishnaidu80@gmail.com

S.V.R Anand
Indian Institute of Science
Bangalore 560012
India

Email: anand@ece.iisc.ernet.in

Malati Hegde
Indian Institute of Science
Bangalore 560012
India

Email: malati@ece.iisc.ernet.in

Charles E. Perkins
Futurewei
2330 Central Expressway
Santa Clara 95050
Unites States

Email: charliep@computer.org

6TiSCH
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

J. Munoz, Ed.
Gridbee Communications - INRIA
E. Riou
Gridbee Communications
D. Barthel
Orange Labs
July 3, 2017

Example Packets for 6TiSCH Configuration
draft-munoz-6tisch-examples-02

Abstract

This draft contains example packets exchanged by nodes implementing the following ietf documents: RFC 8180: Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration, draft-wang-6tisch-6top-protocol-07, RFC 8138: IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) Routing Header and RFC 8025: IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch. All packets are presented both in raw binary and fully parsed contents. This document can be used as a reference when implementing the previous mentioned RFCs and Internet Drafts.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 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. Tools Used	2
2. Network Topology	3
3. Packet Examples	3
3.1. Known Errors in These examples	3
3.2. Enhanced Beacon	3
3.3. RPL DIO	8
3.4. RPL DAO	13
3.4.1. RPL DAO from 2	13
3.4.2. RPL DAO from 3	15
3.5. ACK	19
3.6. ICMPv6 echo request/reply	20
3.6.1. ping 2	20
3.6.2. ping 3	24
3.7. 6Top Commands and Responses	31
4. IANA Considerations	46
5. Security Considerations	46
6. Acknowledgments	46
7. References	46
7.1. Normative References	46
7.2. External Informative References	46
Authors' Addresses	47

1. Tools Used

All results presented in this document are collected by running the OpenWSN firmware [OpenWSN] in simulation mode and capturing the packets exchanged using the development branch of Wireshark. At the time of writing this document, the dissection of the RFC 8138 and RFC 8025 implementation has not been merged into the master branch of Wireshark but it will be by the time of the 1st F-Interop 6TiSCH Interoperability Event.

These are the version of the source code used:

1. Wireshark dissector: <https://github.com/wireshark/wireshark/commit/1aa8ded9a3de8e6fb5b6a7b7dcca9d93bb50dcdd>

2. OpenWSN firmware: <https://github.com/openwsn-berkeley/openwsn-fw/commit/4e05d8bc54b59632d5da771818bf8b4a05b3ce11>
3. OpenWSN software: <https://github.com/openwsn-berkeley/openwsn-sw/commit/e4cdf73cbdbbe88bbe0af48e69abb7cfed84f0d9>

2. Network Topology

Network prefix: bbbb::/64
 MAC address: 14-15-92-cc-00-00-00-0x

```

          PDR=100%          PDR=100%
+-----+          +-----+ +-----+
| x=1 |-----| x=2 |-----| x=3 |
+-----+          +-----+ +-----+
DAGroot

```

3. Packet Examples

3.1. Known Errors in These examples

Looks for "FIXME" in the examples below.

3.2. Enhanced Beacon

Enhanced Beacon sent by 1

== Dissected packet ==

```

IEEE 802.15.4 Enhanced Beacon, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xea40, Frame Type: Beacon, PAN ID Compression, Informa
tion Elements Present, Destination Addressing Mode: Short/16-bit, Frame Version:
  IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .000 = Frame Type: Beacon (0x0)
    .... .0... = Security Enabled: False
    .... .0.... = Frame Pending: False
    .... .0. .... = Acknowledge Request: False
    .... .1.. .... = PAN ID Compression: True
    .... .0 .... = Sequence Number Suppression: False
    .... .1. .... = Information Elements Present: True
    .... 10.. .... = Destination Addressing Mode: Short/16-bit (0x2)
    ..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 90
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Header IEs, Header Termination 1 IE
  Header Termination 1 IE (Payload IEs follow)
    IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0

```

```

0... .. = Type: Header (0)
.011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
.... .. .000 0000 = Length: 0
Payload IE
  MLME IE
    Payload IE TLV: 0x881c, Type: Payload, Id: MLME IE
      1... .. = Type: Payload (1)
      .000 1... .. = Id: MLME IE (0x1)
      .... .000 0001 1100 = Length: 28
    Time Synchronization IE
      Payload Sub IE (short): 0x1a06, Type: Short, Sub Id (Short): TSC
H Synchronization IE
      0... .. = Type: Short (0)
      .001 1010 .... .. = Sub Id (Short): TSCH Synchronization I
E (0x1a)
      .... .. 0000 0110 = Length: 6
      Absolute Slot Number: 10098
      Join Metric: 0
    TSCH Timeslot IE
      Payload IE TLV: 0x1c03, Type: Short, Sub Id (Short): TSCH Timesl
ot IE
      0... .. = Type: Short (0)
      .001 1100 .... .. = Sub Id (Short): TSCH Timeslot IE (0x1c
)
      .... .. 0000 0011 = Length: 3
      Data: 01eb01
    Channel Hopping IE
      Payload Sub IE (long): 0xc801, Type: Long, Sub Id (Long): Channe
l Hopping IE
      1... .. = Type: Long (1)
      .100 1... .. = Sub Id (Long): Channel Hopping IE (0x9
)
      .... .000 0000 0001 = Length: 1
      Hopping Sequence ID: 0x00
    TSCH Slotframe and Link IE
      Payload Sub IE (short): 0x1b0a, Type: Short, Sub Id (Short): TSC
H Slotframe and Link IE
      0... .. = Type: Short (0)
      .001 1011 .... .. = Sub Id (Short): TSCH Slotframe and Lin
k IE (0x1b)
      .... .. 0000 1010 = Length: 10
      Number of Slotframes: 1
      Slotframes [1]
        Slotframe handle: 1
        Slotframe size: 11
        Number of Links: 1
        Link Information
          Timeslot: 0
          Channel Offset: 0
          Link Options: 15
      FCS: 0xfc89 (Correct)

== Raw Bytes ==

0000 40 ea 5a fe ca ff ff 01 00 00 00 cc 92 15 14 00
0010 3f 1c 88 06 1a 72 27 00 00 00 00 03 1c 01 eb 01

```

```
0020 01 c8 00 0a 1b 01 01 0b 00 01 00 00 00 0f 89
0030 fc
```

Enhanced Beacon sent by 2

== Dissected packet ==

```
IEEE 802.15.4 Enhanced Beacon, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xea40, Frame Type: Beacon, PAN ID Compression, Informa
tion Elements Present, Destination Addressing Mode: Short/16-bit, Frame Version:
  IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .000 = Frame Type: Beacon (0x0)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .0... = Acknowledge Request: False
    .... .1... = PAN ID Compression: True
    .... .0... = Sequence Number Suppression: False
    .... .1... = Information Elements Present: True
    .... 10... = Destination Addressing Mode: Short/16-bit (0x2)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 25
  Destination PAN: 0xcafe
  Destination: 0xffff
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    MLME IE
      Payload IE TLV: 0x881c, Type: Payload, Id: MLME IE
      1... .. = Type: Payload (1)
      .000 1... .. = Id: MLME IE (0x1)
      .... .000 0001 1100 = Length: 28
    Time Synchronization IE
      Payload Sub IE (short): 0x1a06, Type: Short, Sub Id (Short): TSC
H Synchronization IE
      0... .. = Type: Short (0)
      .001 1010 .... = Sub Id (Short): TSCH Synchronization I
E (0x1a)
      .... .0000 0110 = Length: 6
      Absolute Slot Number: 10219
      Join Metric: 1
    TSCH Timeslot IE
      Payload IE TLV: 0x1c03, Type: Short, Sub Id (Short): TSCH Timesl
ot IE
      0... .. = Type: Short (0)
      .001 1100 .... = Sub Id (Short): TSCH Timeslot IE (0x1c
)
      .... .0000 0011 = Length: 3
      Data: 01eb01
```

```

        Channel Hopping IE
          Payload Sub IE (long): 0xc801, Type: Long, Sub Id (Long): Channe
l Hopping IE
          1... .... .... .... = Type: Long (1)
          .100 1... .... .... = Sub Id (Long): Channel Hopping IE (0x9
)
          .... .000 0000 0001 = Length: 1
          Hopping Sequence ID: 0x00
        TSCH Slotframe and Link IE
          Payload Sub IE (short): 0x1b0a, Type: Short, Sub Id (Short): TSC
H Slotframe and Link IE
          0... .... .... .... = Type: Short (0)
          .001 1011 .... .... = Sub Id (Short): TSCH Slotframe and Lin
k IE (0x1b)
          .... .... 0000 1010 = Length: 10
          Number of Slotframes: 1
          Slotframes [1]
            Slotframe handle: 1
            Slotframe size: 11
            Number of Links: 1
            Link Information
              Timeslot: 0
              Channel Offset: 0
              Link Options: 15
          FCS: 0xc1b4 (Correct)

```

== Raw Bytes ==

```

0000 40 ea 19 fe ca ff ff 02 00 00 00 cc 92 15 14 00
0010 3f 1c 88 06 1a eb 27 00 00 00 01 03 1c 01 eb 01
0020 01 c8 00 0a 1b 01 01 0b 00 01 00 00 00 0f b4
0030 c1

```

Enhanced Beacon sent by 3

== Dissected packet ==

```

IEEE 802.15.4 Enhanced Beacon, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:03
  Frame Control Field: 0xea40, Frame Type: Beacon, PAN ID Compression, Informa
tion Elements Present, Destination Addressing Mode: Short/16-bit, Frame Version:
  IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .... .... .000 = Frame Type: Beacon (0x0)
    .... .... .... 0... = Security Enabled: False
    .... .... ...0 .... = Frame Pending: False
    .... .... ..0. .... = Acknowledge Request: False
    .... .... .1.. .... = PAN ID Compression: True
    .... ...0 .... .... = Sequence Number Suppression: False
    .... ..1. .... .... = Information Elements Present: True
    .... 10.. .... .... = Destination Addressing Mode: Short/16-bit (0x2)
    ..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 6
  Destination PAN: 0xcafe
  Destination: 0xffff

```

```

Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
Header IEs, Header Termination 1 IE
  Header Termination 1 IE (Payload IEs follow)
    IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
    0... .... = Type: Header (0)
    .011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
    .... .... .000 0000 = Length: 0
Payload IE
  MLME IE
    Payload IE TLV: 0x881c, Type: Payload, Id: MLME IE
    1... .... = Type: Payload (1)
    .000 1... .... = Id: MLME IE (0x1)
    .... .000 0001 1100 = Length: 28
    Time Synchronization IE
      Payload Sub IE (short): 0x1a06, Type: Short, Sub Id (Short): TSC
H Synchronization IE
      0... .... = Type: Short (0)
      .001 1010 .... = Sub Id (Short): TSCH Synchronization I
E (0x1a)
      .... .... 0000 0110 = Length: 6
      Absolute Slot Number: 10417
      Join Metric: 31
      TSCH Timeslot IE
        Payload IE TLV: 0x1c03, Type: Short, Sub Id (Short): TSCH Timesl
ot IE
        0... .... = Type: Short (0)
        .001 1100 .... = Sub Id (Short): TSCH Timeslot IE (0x1c
)
        .... .... 0000 0011 = Length: 3
        Data: 01eb01
        Channel Hopping IE
          Payload Sub IE (long): 0xc801, Type: Long, Sub Id (Long): Channe
l Hopping IE
          1... .... = Type: Long (1)
          .100 1... .... = Sub Id (Long): Channel Hopping IE (0x9
)
          .... .000 0000 0001 = Length: 1
          Hopping Sequence ID: 0x00
          TSCH Slotframe and Link IE
            Payload Sub IE (short): 0x1b0a, Type: Short, Sub Id (Short): TSC
H Slotframe and Link IE
            0... .... = Type: Short (0)
            .001 1011 .... = Sub Id (Short): TSCH Slotframe and Lin
k IE (0x1b)
            .... .... 0000 1010 = Length: 10
            Number of Slotframes: 1
            Slotframes [1]
              Slotframe handle: 1
              Slotframe size: 11
              Number of Links: 1
              Link Information
                Timeslot: 0
                Channel Offset: 0
                Link Options: 15
            FCS: 0x65bf (Correct)

```

== Raw Bytes ==

```
0000  40 ea 06 fe ca ff ff 03 00 00 00 cc 92 15 14 00
0010  3f 1c 88 06 1a b1 28 00 00 00 1f 03 1c 01 eb 01
0020  01 c8 00 0a 1b 01 01 0b 00 01 00 00 00 0f bf
0030  65
```

3.3. RPL DIO

RPL DIO sent by 1

== Dissected packet ==

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:01

Frame Control Field: 0xe841, Frame Type: Data, PAN ID Compression, Destination Addressing Mode: Short/16-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit

```
.... .... .001 = Frame Type: Data (0x1)
.... .... 0... = Security Enabled: False
.... .... .0... = Frame Pending: False
.... .... .0. .... = Acknowledge Request: False
.... .... .1... = PAN ID Compression: True
.... .... .0 .... = Sequence Number Suppression: False
.... .... .0. .... = Information Elements Present: False
.... 10.. .... = Destination Addressing Mode: Short/16-bit (0x2)
..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
```

Sequence Number: 93

Destination PAN: 0xcafe

Destination: 0xffff

Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

FCS: 0x04b3 (Correct)

6LoWPAN

IPHC Header

```
011. .... = Pattern: IP header compression (0x03)
...1 1... .... = Traffic class and flow label: Version, traffic class, and flow label compressed (0x3)
.... .0.. .... = Next header: Inline
.... ..10 .... = Hop limit: 64 (0x2)
.... .... 0... .... = Context identifier extension: False
.... .... .0.. .... = Source address compression: Stateless
.... .... ..11 .... = Source address mode: Compressed (0x0003)
.... .... .... 1... = Multicast address compression: True
.... .... .... .0.. = Destination address compression: Stateless
.... .... .... ..11 = Destination address mode: 8-bits inline (0x0003)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1615:92cc:0:1
Destination: ff02::1a
```

```

Internet Protocol Version 6, Src: fe80::1615:92cc:0:1, Dst: ff02::1a
  0110 .... = Version: 6
  .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, EC
N: Not-ECT)
  .... 0000 00.. .... = Differentiated Services Codepo
int: Default (0)
  .... ..00 .... = Explicit Congestion Notificati
on: Not ECN-Capable Transport (0)
  .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 28
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: fe80::1615:92cc:0:1
Destination: ff02::1a
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
  Type: RPL Control (155)
  Code: 1 (DODAG Information Object)
  Checksum: 0x171b incorrect, should be 0xd255
  [Checksum Status: Bad]
  RPLInstanceID: 0
  Version: 0
  Rank: 256
  Flags: 0x88, Grounded (G), Mode of Operation (MOP): Non-Storing Mode of Oper
ation
    1... .... = Grounded (G): True
    .0.. .... = Zero: False
    ..00 1... = Mode of Operation (MOP): Non-Storing Mode of Operation (0x1)
    .... .000 = DODAG Preference: 0
  Destination Advertisement Trigger Sequence Number (DTSN): 51
  Flags: 0x00
  Reserved: 00
  DODAGID: bbbb::1415:92cc:0:1
== Raw Bytes ==
0000 41 e8 5d fe ca ff ff 01 00 00 00 cc 92 15 14 7a
0010 3b 3a 1a 9b 01 17 1b 00 00 01 00 88 33 00 00 bb
0020 bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01 b3
0030 04

```

RPL DIO sent by 2

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xe841, Frame Type: Data, PAN ID Compression, Destinati
on Addressing Mode: Short/16-bit, Frame Version: IEEE Std 802.15.4-2015, Source
Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... 0... = Security Enabled: False
    .... ..0 .... = Frame Pending: False
    .... ..0. .... = Acknowledge Request: False
    .... .1.. .... = PAN ID Compression: True
    .... .0 .... = Sequence Number Suppression: False

```



```

    .... ..0. .... = Information Elements Present: False
    .... 10.. .... = Destination Addressing Mode: Short/16-bit (0x2)
    ..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 28
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0xfc06 (Correct)
6LoWPAN
  IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1.... .... = Traffic class and flow label: Version, traffic class, and flow label compressed (0x3)
    .... .0.. .... = Next header: Inline
    .... ..10 .... = Hop limit: 64 (0x2)
    .... .... 0... .... = Context identifier extension: False
    .... .... .0.. .... = Source address compression: Stateless
    .... .... ..11 .... = Source address mode: Compressed (0x0003)
    .... .... .... 1... = Multicast address compression: True
    .... .... .... .0.. = Destination address compression: Stateless
    .... .... .... ..11 = Destination address mode: 8-bits inline (0x0003)
    [Source context: fe80::]
    [Destination context: fe80::]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1615:92cc:0:2
    Destination: ff02::1a
Internet Protocol Version 6, Src: fe80::1615:92cc:0:2, Dst: ff02::1a
    0110 .... = Version: 6
    .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 00.. .... = Differentiated Services Codepoint: Default (0)
    .... .... ..00 .... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    .... .... 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 28
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source: fe80::1615:92cc:0:2
    Destination: ff02::1a
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
  Type: RPL Control (155)
  Code: 1 (DODAG Information Object)
  Checksum: 0x161a incorrect, should be 0xd154
  [Checksum Status: Bad]
  RPLInstanceID: 0
  Version: 0
  Rank: 512
  Flags: 0x88, Grounded (G), Mode of Operation (MOP): Non-Storing Mode of Operation

```

```

1... .... = Grounded (G): True
.0.. .... = Zero: False
..00 1... = Mode of Operation (MOP): Non-Storing Mode of Operation (0x1)
.... .000 = DODAG Preference: 0
Destination Advertisement Trigger Sequence Number (DTSN): 51
Flags: 0x00
Reserved: 00
DODAGID: bbbb::1415:92cc:0:1

```

== Raw Bytes ==

```

0000 41 e8 1c fe ca ff ff 02 00 00 00 cc 92 15 14 7a
0010 3b 3a 1a 9b 01 16 1a 00 00 02 00 88 33 00 00 bb
0020 bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01 06
0030 fc

```

RPL DIO sent by 3

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:03
  Frame Control Field: 0xe841, Frame Type: Data, PAN ID Compression, Destination
  Addressing Mode: Short/16-bit, Frame Version: IEEE Std 802.15.4-2015, Source
  Addressing Mode: Long/64-bit
    .... .... .001 = Frame Type: Data (0x1)
    .... .... 0... = Security Enabled: False
    .... .... .0... = Frame Pending: False
    .... .... .0. .... = Acknowledge Request: False
    .... .... .1.. .... = PAN ID Compression: True
    .... .... .0 .... = Sequence Number Suppression: False
    .... .... .0. .... = Information Elements Present: False
    .... 10.. .... = Destination Addressing Mode: Short/16-bit (0x2)
    .... 10.. .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11.. .... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 8
  Destination PAN: 0xcafe
  Destination: 0xffff
  Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
  FCS: 0xab62 (Correct)
6LoWPAN
  IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1... .... = Traffic class and flow label: Version, traffic class, and flow label compressed (0x3)
    .... .0.. .... = Next header: Inline
    .... ..10 .... = Hop limit: 64 (0x2)
    .... .... 0... .... = Context identifier extension: False
    .... .... .0.. .... = Source address compression: Stateless
    .... .... ..11 .... = Source address mode: Compressed (0x0003)
    .... .... .... 1... = Multicast address compression: True
    .... .... .... .0.. = Destination address compression: Stateless

```

```

      .... 11 = Destination address mode: 8-bits inline (0x0003)
      [Source context: fe80::]
      [Destination context: fe80::]
      Next header: ICMPv6 (0x3a)
      Source: fe80::1615:92cc:0:3
      Destination: ff02::1a
Internet Protocol Version 6, Src: fe80::1615:92cc:0:3, Dst: ff02::1a
      0110 .... = Version: 6
      .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, EC
N: Not-ECT)
      .... 0000 00.. .... = Differentiated Services Codepo
int: Default (0)
      .... .... .00 .... = Explicit Congestion Notificati
on: Not ECN-Capable Transport (0)
      .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
      Payload Length: 28
      Next Header: ICMPv6 (58)
      Hop Limit: 64
      Source: fe80::1615:92cc:0:3
      Destination: ff02::1a
      [Source GeoIP: Unknown]
      [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
      Type: RPL Control (155)
      Code: 1 (DODAG Information Object)
      Checksum: 0x1519 incorrect, should be 0xd053
      [Checksum Status: Bad]
      RPLInstanceID: 0
      Version: 0
      Rank: 768
      Flags: 0x88, Grounded (G), Mode of Operation (MOP): Non-Storing Mode of Oper
ation
      1... .... = Grounded (G): True
      .0.. .... = Zero: False
      ..00 1... = Mode of Operation (MOP): Non-Storing Mode of Operation (0x1)
      .... .000 = DODAG Preference: 0
      Destination Advertisement Trigger Sequence Number (DTSN): 51
      Flags: 0x00
      Reserved: 00
      DODAGID: bbbb::1415:92cc:0:1

```

== Raw Bytes ==

```

0000 41 e8 08 fe ca ff ff 03 00 00 00 cc 92 15 14 7a
0010 3b 3a 1a 9b 01 15 19 00 00 03 00 88 33 00 00 bb
0020 bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01 62
0030 ab

```

3.4. RPL DAO

3.4.1. RPL DAO from 2

[RPL DAO from 2] 2->1

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02
 Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit

.... .001 = Frame Type: Data (0x1)
 0... = Security Enabled: False
0... = Frame Pending: False
1... = Acknowledge Request: True
0... = PAN ID Compression: False
0... = Sequence Number Suppression: False
0... = Information Elements Present: False
 11... = Destination Addressing Mode: Long/64-bit (0x3)
 ..10... = Frame Version: IEEE Std 802.15.4-2015 (2)
 11... = Source Addressing Mode: Long/64-bit (0x3)

Sequence Number: 43

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

FCS: 0xe370 (Correct)

6LoWPAN

.... 0001 = Page Number: 1

6LoRH: Routing Protocol Information

100. = Routing Header 6lo: Critical Routing Header (0x04)
 ...0 = Packet direction: UP false, DOWN true: False
 0... = Error detected: False
0... = No link to destination: False
1... = Context identifier extension: True
1 = Context identifier extension: True
 0000 0101 = 6LoRH Type: Routing Protocol Information (0x05)

RPL Instance: 0x00

Sender Rank: 0x02

IPHC Header

011. = Pattern: IP header compression (0x03)
 ...1 1... = Traffic class and flow label: Version, traffic class, and flow label compressed (0x3)
0... = Next header: Inline
10 = Hop limit: 64 (0x2)
 0... = Context identifier extension: False
0... = Source address compression: Stateless
01 = Source address mode: 64-bits inline (0x0001)
 0... = Multicast address compression: False
0... = Destination address compression: Stateless
01 = Destination address mode: 64-bits inline (0x0001)

```

    [Source context: fe80::]
    [Destination context: fe80::]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:2
    Destination: fe80::1415:92cc:0:1
    Internet Protocol Version 6, Src: fe80::1415:92cc:0:2, Dst: fe80::1415:92cc:0:1
    0110 .... = Version: 6
    .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, EC
N: Not-ECT)
    .... 0000 00.. .... = Differentiated Services Codepo
int: Default (0)
    .... .... ..00 .... = Explicit Congestion Notificati
on: Not ECN-Capable Transport (0)
    .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 66
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source: fe80::1415:92cc:0:2
    Destination: fe80::1415:92cc:0:1
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 2 (Destination Advertisement Object)
    Checksum: 0x69d6 incorrect, should be 0xe44b
    [Checksum Status: Bad]
    RPLInstanceID: 0
    Flags: 0x40, DODAGID Present (D)
        0... .... = DAO-ACK Request (K): False
        .1.. .... = DODAGID Present (D): True
        ..00 0000 = Reserved: 0
    Reserved: 00
    DAO Sequence: 0
    DODAGID: bbbb::1415:92cc:0:1
    ICMPv6 RPL Option (RPL Target bbbb::1415:92cc:0:3/128)
        Type: RPL Target (5)
        Length: 18
        Reserved
        Target Length: 128
        Target: bbbb::1415:92cc:0:3
    ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:1)
        Type: Transit Information (6)
        Length: 20
        Flags: 0x00
        Path Control: 0
        Path Sequence: 1
        Path Lifetime: 170
        Parent Address: bbbb::1415:92cc:0:1

```

== Raw Bytes ==

```

0000  21 ec 2b fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 83 05 02 7a 11 3a 14 15 92 cc
0020  00 00 00 02 14 15 92 cc 00 00 00 01 9b 02 69 d6
0030  00 40 00 00 bb bb 00 00 00 00 00 00 14 15 92 cc
0040  00 00 00 01 05 12 00 80 bb bb 00 00 00 00 00 00
0050  14 15 92 cc 00 00 00 03 06 14 00 00 01 aa bb bb
0060  00 00 00 00 00 00 14 15 92 cc 00 00 00 01 70 e3

```

3.4.2. RPL DAO from 3

[RPL DAO from 3] 3->2

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                      Src: 14:15:92:cc:00:00:00:03
  Frame Control Field: 0xec21, Frame Type: Data
    .... .001 = Frame Type: Data (0x0001)
    .... .0... = Security Enabled: False
    .... .0 .... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0.. .... = Intra-PAN: False
    .... .0 .... = Sequence Number Suppression: False
    .... .0. .... = Information Elements present: False
    .... 11.. .... = Destination Addressing Mode:
                      Long/64-bit (0x0003)
    ..10 .... = Frame Version: 2
    11.. .... = Source Addressing Mode:
                      Long/64-bit (0x0003)

  Sequence Number: 5
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
  FCS: 0x1640 (Correct)
6LoWPAN
  .... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
  100. .... = Routing Header 6lo: Critical Routing Header (0x04)
  ...0 .... = Packet direction:
                UP false, DOWN true: False
  .... 0... = Error detected: False
  .... .0.. = No link to destination: False
  .... .1. = Context identifier extension: True
  .... .1. = Context identifier extension: True
  .... 0000 0101 = 6LoRH Type: Routing Protocol Information
  RPL Instance: 0x00
  Sender Rank: 0x21
IPHC Header

```

```

011. .... = Pattern: IP header compression (0x03)
...1 1... .... = Traffic class and flow label: Version,
                traffic class, and flow label
                compressed (0x0003)
.... .0... .... = Next header: Inline
.... ..10 .... = Hop limit: 64 (0x0002)
.... .... 0... .... = Context identifier extension: False
.... .... .0... .... = Source address compression: Stateless
.... .... ..01 .... = Source address mode: 64-bits inline(0x01)
.... .... .... 0... = Multicast address compression: False
.... .... .... .0... = Dest address compression: Stateless
.... .... .... ..01 = Dest address mode: 64-bits inline (0x01)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1415:92cc:0:3
Destination: fe80::1415:92cc:0:1
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3,
                                Dst: fe80::1415:92cc:0:1

0110 .... = Version: 6
.... 0000 0000 .... = Traffic class:
                                0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated
                                Services Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
                                Notification: Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 46
Next header: ICMPv6 (58)
Hop limit: 64
Source: fe80::1415:92cc:0:3
Destination: fe80::1415:92cc:0:1
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 2 (Destination Advertisement Object)
Checksum: 0xd31a [incorrect, should be 0x4d90]
RPLInstanceID: 0
Flags: 0x40
    0... .... = DAO-ACK Request (K): False
    .1... .... = DODAGID Present (D): True
    ..00 0000 = Reserved: 0
Reserved: 00
DAO Sequence: 0
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:2)
    Type: Transit Information (6)
    Length: 20
    Flags: 0x00

```

```

    0... .... = External: Not set
    .000 0000 = Reserved: 0
    Path Control: 0
    Path Sequence: 0
    Path Lifetime: 170
    Parent Address: bbbb::1415:92cc:0:2

```

== Raw Bytes ==

```

0000  21 ec 05 fe ca 02 00 00 00 cc 92 15 14 03 00 00
0010  00 cc 92 15 14 f1 83 05 21 7a 11 3a 14 15 92 cc
0020  00 00 00 03 14 15 92 cc 00 00 00 01 9b 02 d3 1a
0030  00 40 00 00 bb bb 00 00 00 00 00 00 14 15 92 cc
0040  00 00 00 01 06 14 00 00 00 aa bb bb 00 00 00 00
0050  00 00 14 15 92 cc 00 00 00 02 40 16

```

[RPL DAO from 3] 2->1

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
                      Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xec21, Frame Type: Data
    .... .... .... .001 = Frame Type: Data (0x0001)
    .... .... .... 0... = Security Enabled: False
    .... .... ...0 .... = Frame Pending: False
    .... .... ..1. .... = Acknowledge Request: True
    .... .... .0.. .... = Intra-PAN: False
    .... ...0 .... .... = Sequence Number Suppression: False
    .... ..0. .... .... = Information Elements present: False
    .... 11.. .... .... = Destination Addressing Mode:
                          Long/64-bit (0x0003)
    ..10 .... .... .... = Frame Version: 2
    11.. .... .... .... = Source Addressing Mode:
                          Long/64-bit (0x0003)

  Sequence Number: 11
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  FCS: 0x2135 (Correct)
6LoWPAN
  .... 0001 = Page Number: 1
  6LoRH: Routing Protocol Information
    100. .... = Routing Header 6lo: Critical Routing Header (0x04)
    ...0 .... .... .... = Packet direction:
                          UP false, DOWN true: False
    .... 0... .... .... = Error detected: False
    .... .0.. .... .... = No link to destination: False

```



```

.....1. .... = Context identifier extension: True
....1 .... = Context identifier extension: True
.... 0000 0101 = 6loRH Type: Routing Protocol Information
RPL Instance: 0x00
Sender Rank: 0x03
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... .... = Traffic class and flow label: Version,
                  traffic class, and flow label
                  compressed (0x0003)
.....0... .... = Next header: Inline
.....10 .... = Hop limit: 64 (0x0002)
..... 0... .... = Context identifier extension: False
..... 0... .... = Source address compression: Stateless
..... 01 .... = Source address mode: 64-bits inline (0x01)
..... 0... .... = Multicast address compression: False
..... 0... .... = Dest address compression: Stateless
..... 01 .... = Dest address mode: 64-bits inline (0x01)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1415:92cc:0:3
Destination: fe80::1415:92cc:0:1
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3,
                               Dst: fe80::1415:92cc:0:1
0110 .... = Version: 6
.... 0000 0000 .... = Traffic class:
                               0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated
                               Services Codepoint: Default (0)
.... 0000 ..00 .... = Explicit Congestion
                               Notification: Not ECN-Capable Transport (0)
.... 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 46
Next header: ICMPv6 (58)
Hop limit: 64
Source: fe80::1415:92cc:0:3
Destination: fe80::1415:92cc:0:1
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 2 (Destination Advertisement Object)
Checksum: 0xd31a [incorrect, should be 0x4d90]
RPLInstanceID: 0
Flags: 0x40
0... .... = DAO-ACK Request (K): False
.1... .... = DODAGID Present (D): True
..00 0000 = Reserved: 0
Reserved: 00

```

```
DAO Sequence: 0
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:2)
  Type: Transit Information (6)
  Length: 20
  Flags: 0x00
    0... .... = External: Not set
    .000 0000 = Reserved: 0
  Path Control: 0
  Path Sequence: 0
  Path Lifetime: 170
  Parent Address: bbbb::1415:92cc:0:2
```

== Raw Bytes ==

```
0000  21 ec 0b fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 83 05 03 7a 11 3a 14 15 92 cc
0020  00 00 00 03 14 15 92 cc 00 00 00 01 9b 02 d3 1a
0030  00 40 00 00 bb bb 00 00 00 00 00 00 14 15 92 cc
0040  00 00 00 01 06 14 00 00 00 aa bb bb 00 00 00 00
0050  00 00 14 15 92 cc 00 00 00 02 35 21
```

3.5. ACK

ACK

== Dissected packet ==

IEEE 802.15.4 Ack, Sequence Number: 69, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02

Frame Control Field: 0xee02, Frame Type: Ack, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit

```

.... .... .010 = Frame Type: Ack (0x2)
.... .... 0... = Security Enabled: False
.... .... .0... = Frame Pending: False
.... .... .0... = Acknowledge Request: False
.... .... .0... = PAN ID Compression: False
.... .0... = Sequence Number Suppression: False
.... .1... = Information Elements Present: True
.... 11... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11... .. = Source Addressing Mode: Long/64-bit (0x3)

```

Sequence Number: 69

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Header IEs, Time Correction IE

Time Correction IE

IE Header: 0x0f02, Type: Header, Id: Time Correction IE, Length: 2

0... = Type: Header (0)

.000 1111 0... = Id: Time Correction IE (0x1e)

....000 0010 = Length: 2

Time Sync Info: 0x0000, Time Correction: 0, Nack: Acknowledgement

.... 0000 0000 0000 = Time Correction: 0[micro]s

0... = Nack: Acknowledgement

FCS: 0xaa9c (Correct)

== Raw Bytes ==

```

0000 02 ee 45 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 02 0f 00 00 9c aa

```

3.6. ICMPv6 echo request/reply

3.6.1. ping 2

[ping 2] ICMPv6 echo request 1->2

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,

Src: 14:15:92:cc:00:00:00:01

Frame Control Field: 0xec21, Frame Type: Data

....001 = Frame Type: Data (0x0001)

```

..... 0... = Security Enabled: False
..... 0... = Frame Pending: False
..... 1... = Acknowledge Request: True
..... 0... = Intra-PAN: False
..... 0... = Sequence Number Suppression: False
..... 0... = Information Elements present: False
..... 11.. = Destination Addressing Mode:
                Long/64-bit (0x0003)
..10 ..... = Frame Version: 2
11.. ..... = Source Addressing Mode:
                Long/64-bit (0x0003)

Sequence Number: 42
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
FCS: 0xd916 (Correct)
6LoWPAN
.... 0001 = Page Number: 1
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... = Traffic class and flow label: Version,
                traffic class, and flow label
                compressed (0x0003)
..... 0... = Next header: Inline
..... 00 .. = Hop limit: Inline (0x0000)
..... 0... = Context identifier extension: False
..... 0... = Source address compression: Stateless
..... 00 .. = Source address mode: Inline (0x0000)
..... 0... = Multicast address compression: False
..... 0... = Dest address compression: Stateless
..... 00 .. = Dest address mode: Inline (0x0000)
Next header: ICMPv6 (0x3a)
Hop limit: 64
Source: bbbb::1
Destination: bbbb::1415:92cc:0:2
Internet Protocol Version 6, Src: bbbb::1, Dst: bbbb::1415:92cc:0:2
0110 .... = Version: 6
.... 0000 0000 ..... = Traffic class: 0x00
                        (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. ..... = Differentiated
                        Services Codepoint: Default (0)
.... 00 ..00 ..... = Explicit Congestion
                        Notification: Not ECN-Capable Transport (0)
.... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 64
Next header: ICMPv6 (58)
Hop limit: 64
Source: bbbb::1

```

```

    Destination: bbbb::1415:92cc:0:2
Internet Control Message Protocol v6
  Type: Echo (ping) request (128)
  Code: 0
  Checksum: 0xf7be [correct]
  Identifier: 0x47c5
  Sequence: 1
  [No response seen]
  Data (56 bytes)

```

```

0000  d0 27 b2 56 00 00 00 00 d5 52 0b 00 00 00 00 00
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
0030  30 31 32 33 34 35 36 37
      Data: d027b25600000000d5520b0000000001011121314151617...
      [Length: 56]

```

```

== Raw Bytes ==

```

```

0000  21 ec 2a fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 f1 78 00 3a 40 bb bb 00 00 00 00
0020  00 00 00 00 00 00 00 00 00 00 01 bb bb 00 00 00 00
0030  00 00 14 15 92 cc 00 00 00 02 80 00 f7 be 47 c5
0040  00 01 d0 27 b2 56 00 00 00 00 d5 52 0b 00 00 00
0050  00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d
0060  1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d
0070  2e 2f 30 31 32 33 34 35 36 37 16 d9

```

```

[ping 2] ICMPv6 echo reply 2->1

```

```

== Dissected packet ==

```

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
                      Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xec21, Frame Type: Data
  .... .001 = Frame Type: Data (0x0001)
  .... .0... = Security Enabled: False
  .... .0... = Frame Pending: False
  .... .1. .... = Acknowledge Request: True
  .... .0.. .... = Intra-PAN: False
  .... .0 .... = Sequence Number Suppression: False
  .... .0. .... = Information Elements present: False
  .... 11.. .... = Destination Addressing Mode:
                      Long/64-bit (0x0003)
  ..10 .... = Frame Version: 2
  11.. .... = Source Addressing Mode:
                      Long/64-bit (0x0003)
Sequence Number: 48

```

```

Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0xe105 (Correct)
6LoWPAN
.... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
100. .... = Routing Header 6lo: Critical Routing Header (0x04)
...0 .... = Packet direction:
          UP false, DOWN true: False
.... 0... .... = Error detected: False
.... .0.. .... = No link to destination: False
.... ..1. .... = Context identifier extension: True
.... ...0 .... = Context identifier extension: False
.... .... 0000 0101 = 6LoRH Type: Routing Protocol Information
RPL Instance: 0x00
Sender Rank: 0x0338
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... .... = Traffic class and flow label: Version,
                traffic class, and flow label
                compressed (0x0003)
.... .0.. .... = Next header: Inline
.... ..10 .... = Hop limit: 64 (0x0002)
.... .... 0... = Context identifier extension: False
.... .... .0.. = Source address compression: Stateless
.... .... ..01 = Source address mode: 64-bits inline (0x01)
.... .... .... 0... = Multicast address compression: False
.... .... .... .0.. = Dest address compression: Stateless
.... .... .... ..01 = Dest address mode: 64-bits inline (0x01)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1415:92cc:0:2
Destination: fe80::1
Internet Protocol Version 6, Src: fe80::1415:92cc:0:2, Dst: fe80::1
0110 .... = Version: 6
.... 0000 0000 .... = Traffic class:
                0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated
                Services Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
                Notification: Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 64
Next header: ICMPv6 (58)
Hop limit: 64
Source: fe80::1415:92cc:0:2

```

```

    Destination: fe80::1
Internet Control Message Protocol v6
  Type: Echo (ping) reply (129)
  Code: 0
  Checksum: 0xf6be [incorrect, should be 0x7134]
  Identifier: 0x47c5
  Sequence: 1
  Data (56 bytes)

0000  d0 27 b2 56 00 00 00 00 d5 52 0b 00 00 00 00 00
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
0030  30 31 32 33 34 35 36 37
      Data: d027b25600000000d5520b0000000001011121314151617...
      [Length: 56]

```

== Raw Bytes ==

```

0000  21 ec 30 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 82 05 03 38 7a 11 3a 14 15 92
0020  cc 00 00 00 02 00 00 00 00 00 00 00 01 81 00 f6
0030  be 47 c5 00 01 d0 27 b2 56 00 00 00 00 d5 52 0b
0040  00 00 00 00 00 10 11 12 13 14 15 16 17 18 19 1a
0050  1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a
0060  2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 05 e1

```

3.6.2. ping 3

[ping 3] ICMPv6 echo request 1->2

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                      Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xec21, Frame Type: Data
    .... .001 = Frame Type: Data (0x0001)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0.. .... = Intra-PAN: False
    .... .0 .... = Sequence Number Suppression: False
    .... .0. .... = Information Elements present: False
    .... 11.. .... = Destination Addressing Mode:
                      Long/64-bit (0x03)
    ..10 .... .... = Frame Version: 2
    11.. .... .... = Source Addressing Mode:
                      Long/64-bit (0x03)

  Sequence Number: 34

```

```

Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
FCS: 0x0366 (Correct)
6LoWPAN
.... 0001 = Page Number: 1
6LoRH: Routing Header 3, 8 byte compression
100. .... = Routing Header 6lo: Critical Routing Header (0x04)
...0 0000 .... = 6LoRH Hop Number - 1: 0x0000
.... 0000 0011 = 6LoRH Type: Routing Header 3,
               8 byte compression (0x0003)
Source/8, Delta: ::1415:92cc:0:2
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... .... = Traffic class and flow label: Version,
               traffic class, and flow label
               compressed (0x0003)
.... .0.. .... = Next header: Inline
.... ..00 .... = Hop limit: Inline (0x0000)
.... .... 0... = Context identifier extension: False
.... .... .0.. = Source address compression: Stateless
.... .... ..00 = Source address mode: Inline (0x0000)
.... .... .... 0... = Multicast address compression: False
.... .... .... .0.. = Dest address compression: Stateless
.... .... .... ..00 = Dest address mode: Inline (0x00)
Next header: ICMPv6 (0x3a)
Hop limit: 64
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
Internet Protocol Version 6, Src: bbbb::1, Dst: bbbb::1415:92cc:0:3
0110 .... = Version: 6
.... 0000 0000 .... = Traffic class:
               0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated
               Services Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
               Notification: Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 18
Next header: ICMPv6 (58)
Hop limit: 64
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0x13f9 [correct]
Identifier: 0x3943

```


Sequence: 1

Data (10 bytes)

```
0000  00 01 02 03 04 05 06 07 08 09
      Data: 00010203040506070809
      [Length: 10]
```

== Raw Bytes ==

```
0000  21 ec 22 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 f1 80 03 14 15 92 cc 00 00 00 02
0020  78 00 3a 40 bb bb 00 00 00 00 00 00 00 00 00 00
0030  00 00 00 01 bb bb 00 00 00 00 00 00 14 15 92 cc
0040  00 00 00 03 80 00 13 f9 39 43 00 01 00 01 02 03
0050  04 05 06 07 08 09 66 03
```

[ping 3] ICMPv6 echo request 2->3

== Dissected packet ==

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:03,
      Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xec21, Frame Type: Data
    .... .001 = Frame Type: Data (0x0001)
    .... .0... = Security Enabled: False
    .... .0.... = Frame Pending: False
    .... .1.... = Acknowledge Request: True
    .... .0... = Intra-PAN: False
    .... .0.... = Sequence Number Suppression: False
    .... .0.... = Information Elements present: False
    .... 11... = Destination Addressing Mode:
                  Long/64-bit (0x03)
    ..10 .... = Frame Version: 2
    11... .. = Source Addressing Mode:
                  Long/64-bit (0x03)

  Sequence Number: 35
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  FCS: 0x793f (Correct)
6LoWPAN
  IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1... = Traffic class and flow label: Version,
                  traffic class, and flow label
                  compressed (0x0003)
    .... .0... = Next header: Inline
```

```

..... ..00 ..... = Hop limit: Inline (0x0000)
..... ..00 ..... = Context identifier extension: False
..... ..00 ..... = Source address compression: Stateless
..... ..00 ..... = Source address mode: Inline (0x0000)
..... ..00 ..... = Multicast address compression: False
..... ..00 ..... = Dest address compression: Stateless
..... ..00 ..... = Dest address mode: Inline (0x0000)
Next header: ICMPv6 (0x3a)
Hop limit: 64
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
Internet Protocol Version 6, Src: bbbb::1, Dst: bbbb::1415:92cc:0:3
0110 ..... = Version: 6
..... 0000 0000 ..... = Traffic class:
                        0x00 (DSCP: CS0, ECN: Not-ECT)
..... 0000 00.. ..... = Differentiated
                        Services Codepoint: Default (0)
..... ..00 ..... = Explicit Congestion
                        Notification: Not ECN-Capable Transport (0)
..... ..00 0000 0000 0000 0000 0000 =Flowlabel: 0x00000000
Payload length: 18
Next header: ICMPv6 (58)
Hop limit: 64
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0x13f9 [correct]
Identifier: 0x3943
Sequence: 1

Data (10 bytes)

0000 00 01 02 03 04 05 06 07 08 09
      Data: 00010203040506070809
      [Length: 10]

== Raw Bytes ==

0000 21 ec 23 fe ca 03 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 78 00 3a 40 bb bb 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 01 bb bb 00 00 00 00 00
0030 00 14 15 92 cc 00 00 00 03 80 00 13 f9 39 43 00
0040 01 00 01 02 03 04 05 06 07 08 09 3f 79

[ping 3] ICMPv6 echo reply 3->2

```

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
      Src: 14:15:92:cc:00:00:00:03
  Frame Control Field: 0xec21, Frame Type: Data
    .... .001 = Frame Type: Data (0x0001)
    .... .0... = Security Enabled: False
    .... .0 .... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0.. .... = Intra-PAN: False
    .... .0 .... = Sequence Number Suppression: False
    .... .0. .... = Information Elements present: False
    .... 11.. .... = Destination Addressing Mode:
                      Long/64-bit (0x03)
    ..10 .... = Frame Version: 2
    11.. .... = Source Addressing Mode:
                      Long/64-bit (0x03)

  Sequence Number: 23
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
  FCS: 0x84f7 (Correct)
6LoWPAN
  .... 0001 = Page Number: 1
  6LoRH: Routing Protocol Information
    100. .... = Routing Header 6lo: Critical Routing Header (0x04)
    ...0 .... = Packet direction:
                UP false, DOWN true: False
    .... 0... .... = Error detected: False
    .... .0.. .... = No link to destination: False
    .... .1. .... = Context identifier extension: True
    .... .1 .... = Context identifier extension: True
    .... 0000 0101 = 6LoRH Type: Routing Protocol Information
  RPL Instance: 0x00
  Sender Rank: 0x07
  IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1... .... = Traffic class and flow label: Version,
                      traffic class, and flow label
                      compressed (0x03)
    .... .0.. .... = Next header: Inline
    .... .10 .... = Hop limit: 64 (0x0002)
    .... .0 .... = Context identifier extension: False
    .... .0.. .... = Source address compression: Stateless
    .... .01 .... = Source address mode: 64-bits inline (0x01)
    .... .0 .... = Multicast address compression: False
    .... .0.. .... = Dest address compression: Stateless
    .... .01 .... = Dest address mode: 64-bits inline (0x01)

```

```

    [Source context: fe80::]
    [Destination context: fe80::]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:3
    Destination: fe80::1
    Internet Protocol Version 6, Src: fe80::1415:92cc:0:3, Dst: fe80::1
    0110 .... = Version: 6
    .... 0000 0000 .... = Traffic class:
        0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 00.. .... = Differentiated
        Services Codepoint: Default (0)
    .... .... ..00 .... = Explicit Congestion
        Notification: Not ECN-Capable Transport (0)
    .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 18
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1415:92cc:0:3
    Destination: fe80::1
    Internet Control Message Protocol v6
    Type: Echo (ping) reply (129)
    Code: 0
    Checksum: 0x12f9 [incorrect, should be 0x8d6e]
    [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect]
    Identifier: 0x3943
    Sequence: 1
    Data (10 bytes)

0000  00 01 02 03 04 05 06 07 08 09
      Data: 00010203040506070809
      [Length: 10]

```

== Raw Bytes ==

```

0000  21 ec 17 fe ca 02 00 00 00 cc 92 15 14 03 00 00
0010  00 cc 92 15 14 f1 83 05 07 7a 11 3a 14 15 92 cc
0020  00 00 00 03 00 00 00 00 00 00 00 01 81 00 12 f9
0030  39 43 00 01 00 01 02 03 04 05 06 07 08 09 f7 84

```

[ping 3] ICMPv6 echo reply 2->1

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
      Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xec21, Frame Type: Data
      .... .001 = Frame Type: Data (0x0001)
      .... .0... = Security Enabled: False

```

```

.....0..... = Frame Pending: False
.....1..... = Acknowledge Request: True
.....0..... = Intra-PAN: False
.....0..... = Sequence Number Suppression: False
.....0..... = Information Elements present: False
.....11..... = Destination Addressing Mode:
                  Long/64-bit (0x03)
..10..... = Frame Version: 2
11..... = Source Addressing Mode:
                  Long/64-bit (0x03)

Sequence Number: 36
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0x7dbc (Correct)
6LoWPAN
.... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
100. .... = Routing Header 6lo: Critical Routing Header (0x04)
...0..... = Packet direction:
              UP false, DOWN true: False
.... 0... .. = Error detected: False
.... .0... .. = No link to destination: False
.... .1. .... = Context identifier extension: True
.... ..1 .... = Context identifier extension: True
.... .. 0000 0101 = 6LoRH Type: Routing Protocol Information
RPL Instance: 0x00
Sender Rank: 0x03
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... .. = Traffic class and flow label: Version,
              traffic class, and flow label
              compressed (0x0003)
.... .0... .. = Next header: Inline
.... ..10 .... = Hop limit: 64 (0x0002)
.... .. 0... .. = Context identifier extension: False
.... .. .0... .. = Source address compression: Stateless
.... .. ..01 .... = Source address mode: 64-bits inline (0x01)
.... .. .. 0... = Multicast address compression: False
.... .. .. .0... = Dest address compression: Stateless
.... .. .. ..01 = Dest address mode: 64-bits inline (0x01)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1415:92cc:0:3
Destination: fe80::1
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3, Dst: fe80::1
0110 .... = Version: 6

```

```

..... 0000 0000 ..... = Traffic class:
                                0x00 (DSCP: CS0, ECN: Not-ECT)
..... 0000 00.. ..... = Differentiated
                                Services Codepoint: Default (0)
..... ..00 ..... = Explicit Congestion
                                Notification: Not ECN-Capable Transport (0)
..... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
Payload length: 18
Next header: ICMPv6 (58)
Hop limit: 64
Source: fe80::1415:92cc:0:3
Destination: fe80::1
Internet Control Message Protocol v6
Type: Echo (ping) reply (129)
Code: 0
Checksum: 0x12f9 [incorrect, should be 0x8d6e]
    [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect]
Identifier: 0x3943
Sequence: 1
Data (10 bytes)

```

```

0000 00 01 02 03 04 05 06 07 08 09
      Data: 00010203040506070809
      [Length: 10]

```

== Raw Bytes ==

```

0000 21 ec 24 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 f1 83 05 03 7a 11 3a 14 15 92 cc
0020 00 00 00 03 00 00 00 00 00 00 00 01 81 00 12 f9
0030 39 43 00 01 00 01 02 03 04 05 06 07 08 09 bc 7d

```

3.7. 6Top Commands and Responses

6top Command ADD 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
..... .001 = Frame Type: Data (0x1)
..... 0... = Security Enabled: False
..... ..0 ..... = Frame Pending: False
..... ..1. .... = Acknowledge Request: True
..... ..0.. .... = PAN ID Compression: False
..... ..0 ..... = Sequence Number Suppression: False
..... ..1. .... = Information Elements Present: True
..... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 ..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. ..... = Source Addressing Mode: Long/64-bit (0x3)

```

```

Sequence Number: 224
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
        IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
    0... .... = Type: Header (0)
    .011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
    .... .... .000 0000 = Length: 0
Payload IE
    Payload IE, IETF IE, Length: 21
    Payload IE TLV: 0xa815, Type: Payload, Id: IETF IE
        1... .... = Type: Payload (1)
        .010 1... .... = Id: IETF IE (0x5)
        .... .000 0001 0101 = Length: 21
    Sub-ID: 201
    6top IE
        .... 0000 = 6P Version: 0
        ..00 .... = Type: Request (0x0)
        00.. .... = Reserved: 0x0
        Code: 0x01 (ADD)
        SFID (6top Scheduling Function ID): 0x00
        .... 0000 = SeqNum: 0
        0000 .... = GEN: Clear (0)
        Metadata: 0x0100
        Cell Options: TX (0x01)
            .... ...1 = Transmit (TX) Cell: 0x1
            .... ..0. = Receive (RX) Cell: 0x0
            .... .0.. = SHARED Cell: 0x0
            0000 0... = Reserved: 0x00
        Number of Cells: 1
        CellList
            Cell: 08000200
                Slot Offset: 0x0008
                Channel Offset: 0x0002
            Cell: 07000200
                Slot Offset: 0x0007
                Channel Offset: 0x0002
            Cell: 06000200
                Slot Offset: 0x0006
                Channel Offset: 0x0002
        FCS: 0x455a (Correct)
== Raw Bytes ==

0000 21 ee e0 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 00 3f 15 a8 c9 00 01 00 00 00 01
0020 01 01 08 00 02 00 07 00 02 00 06 00 02 00 5a 45

```

6top Response to ADD 1->2


```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1... = Information Elements Present: True
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 210
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 9
      Payload IE TLV: 0xa809, Type: Payload, Id: IETF IE
        1... .. = Type: Payload (1)
        .010 1... .. = Id: IETF IE (0x5)
        .... .000 0000 1001 = Length: 9
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..01 .... = Type: Response (0x1)
        00... .. = Reserved: 0x0
        Code: 0x00 (SUCCESS)
        SFID (6top Scheduling Function ID): 0x00
        .... 0000 = SeqNum: 0
        0000 .... = GEN: Clear (0)
        CellList
          Cell: 08000200
          Slot Offset: 0x0008
          Channel Offset: 0x0002
      FCS: 0xa534 (Correct)

== Raw Bytes ==

0000 21 ee d2 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 09 a8 c9 10 00 00 00 08 00
0020 02 00 34 a5

```

6top Command COUNT 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1. .... = Information Elements Present: True
    .... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 125
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 8
      Payload IE TLV: 0xa808, Type: Payload, Id: IETF IE
        1... .. = Type: Payload (1)
        .010 1... .. = Id: IETF IE (0x5)
        .... .000 0000 1000 = Length: 8
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..00 .... = Type: Request (0x0)
        00.. .... = Reserved: 0x0
        Code: 0x04 (COUNT)
        SFID (6top Scheduling Function ID): 0x00
        .... 0001 = SeqNum: 1
        0001 .... = GEN: Lollipop Counter Value (1)
        Metadata: 0x0000
        Cell Options: TX (0x01)
          .... .1 = Transmit (TX) Cell: 0x1
          .... .0. = Receive (RX) Cell: 0x0
          .... .0.. = SHARED Cell: 0x0
          0000 0... = Reserved: 0x00
      FCS: 0x0e78 (Correct)

== Raw Bytes ==

```

```
0000  21 ee 7d fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 00 3f 08 a8 c9 00 04 00 11 00 00
0020  01 78 0e
```

6top Response to COUNT 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1. .... = Information Elements Present: True
    .... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 99
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .... = Type: Header (0)
      .011 1111 0... = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 7
      Payload IE TLV: 0xa807, Type: Payload, Id: IETF IE
        1... .... = Type: Payload (1)
        .010 1... = Id: IETF IE (0x5)
        .... .000 0000 0111 = Length: 7
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..01 .... = Type: Response (0x1)
        00.. .... = Reserved: 0x0
        Code: 0x00 (SUCCESS)
        SFID (6top Scheduling Function ID): 0x00
        .... 0001 = SeqNum: 1
        0001 .... = GEN: Lollipop Counter Value (1)
        Total Number of Cells: 1
      FCS: 0x711c (Correct)
  == Raw Bytes ==

0000 21 ee 63 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 07 a8 c9 10 00 00 11 01 00
0020 1c 71

```

6top Command DELETE 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1... = Information Elements Present: True
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 174
  Destination PAN: 0xcfe
  Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... = Type: Header (0)
      .011 1111 0... = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 13
    Payload IE TLV: 0xa80d, Type: Payload, Id: IETF IE
      1... = Type: Payload (1)
      .010 1... = Id: IETF IE (0x5)
      .... .000 0000 1101 = Length: 13
    Sub-ID: 201
    6top IE
      .... 0000 = 6P Version: 0
      ..00 .... = Type: Request (0x0)
      00... = Reserved: 0x0
      Code: 0x02 (DELETE)
      SFID (6top Scheduling Function ID): 0x00
      .... 0011 = SeqNum: 3
      0001 .... = GEN: Lollipop Counter Value (1)
      Metadata: 0x0000
      Cell Options: TX (0x01)
        .... .1 = Transmit (TX) Cell: 0x1
        .... .0 = Receive (RX) Cell: 0x0
        .... .0... = SHARED Cell: 0x0
        0000 0... = Reserved: 0x00
      Number of Cells: 1
      CellList
        Cell: 08000200
          Slot Offset: 0x0008
          Channel Offset: 0x0002

```

FCS: 0x99e7 (Correct)
== Raw Bytes ==

```
0000  21 ee ae fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 00 3f 0d a8 c9 00 02 00 13 00 00
0020  01 01 08 00 02 00 e7 99
```

6top Response to DELETE 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0.... = Frame Pending: False
    .... .1.... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0.... = Sequence Number Suppression: False
    .... .1.... = Information Elements Present: True
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 88
  Destination PAN: 0xcfe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 9
      Payload IE TLV: 0xa809, Type: Payload, Id: IETF IE
        1... .. = Type: Payload (1)
        .010 1... .. = Id: IETF IE (0x5)
        .... .000 0000 1001 = Length: 9
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..01 .... = Type: Response (0x1)
        00.. .... = Reserved: 0x0
        Code: 0x00 (SUCCESS)
        SFID (6top Scheduling Function ID): 0x00
        .... 0011 = SeqNum: 3
        0001 .... = GEN: Lollipop Counter Value (1)
        CellList
          Cell: 08000200
            Slot Offset: 0x0008
            Channel Offset: 0x0002
      FCS: 0x23c5 (Correct)
  == Raw Bytes ==

0000 21 ee 58 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 09 a8 c9 10 00 00 13 08 00
0020 02 00 c5 23

```

6top Command RELOCATE 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... = Frame Type: Data (0x1)
    ....0... = Security Enabled: False
    ....0... = Frame Pending: False
    ....1... = Acknowledge Request: True
    ....0... = PAN ID Compression: False
    ....0... = Sequence Number Suppression: False
    ....1... = Information Elements Present: True
    ....11... = Destination Addressing Mode: Long/64-bit (0x3)
    ....10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    ....11... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 218
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... = Type: Header (0)
      .011 1111 0... = Id: Header Termination 1 IE (0x7e)
      ....0000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 29
      Payload IE TLV: 0xa81d, Type: Payload, Id: IETF IE
        1... = Type: Payload (1)
        .010 1... = Id: IETF IE (0x5)
        ....0000 0001 1101 = Length: 29
      Sub-ID: 201
      6top IE
        ....0000 = 6P Version: 0
        ..00... = Type: Request (0x0)
        00... = Reserved: 0x0
        Code: 0x03 (RELOCATE)
        SFID (6top Scheduling Function ID): 0x00
        ....1001 = SeqNum: 9
        0100... = GEN: Lollipop Counter Value (4)
        Metadata: 0xc400
        Cell Options: TX (0x01)
          ....1... = Transmit (TX) Cell: 0x1
          ....0... = Receive (RX) Cell: 0x0
          ....0... = SHARED Cell: 0x0
          0000 0... = Reserved: 0x00
        Number of Cells: 2
        Rel. CellList
        Cand. CellList

```


FCS: 0xbae5 (Correct)
 == Raw Bytes ==

```
0000 21 ee da fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 00 3f 1d a8 c9 00 03 00 49 00 c4
0020 01 02 0a 00 02 00 09 00 02 00 05 00 02 00 08 00
0030 02 00 07 00 02 00 e5 ba
```

6top Response to RELOCATE 1->2

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1... = Information Elements Present: True
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 245
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 13
      Payload IE TLV: 0xa80d, Type: Payload, Id: IETF IE
        1... .. = Type: Payload (1)
        .010 1... .. = Id: IETF IE (0x5)
        .... .000 0000 1101 = Length: 13
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..01 .... = Type: Response (0x1)
        00.. .... = Reserved: 0x0
        Code: 0x00 (SUCCESS)
        SFID (6top Scheduling Function ID): 0x00
        .... 1001 = SeqNum: 9
        0100 .... = GEN: Lollipop Counter Value (4)
        CellList
```

```
      Cell: 08000200
            Slot Offset: 0x0008
            Channel Offset: 0x0002
      Cell: 05000200
            Slot Offset: 0x0005
            Channel Offset: 0x0002
      FCS: 0xd405 (Correct)
== Raw Bytes ==

0000  21 ee f5 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 00 3f 0d a8 c9 10 00 00 49 08 00
0020  02 00 05 00 02 00 05 d4
```

6top Command CLEAR 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01, Src: 14:15:92:cc:00:00:00:02
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1. .... = Information Elements Present: True
    .... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 156
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 7
      Payload IE TLV: 0xa807, Type: Payload, Id: IETF IE
        1... .. = Type: Payload (1)
        .010 1... .. = Id: IETF IE (0x5)
        .... .000 0000 0111 = Length: 7
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..00 .... = Type: Request (0x0)
        00.. .... = Reserved: 0x0
        Code: 0x06 (CLEAR)
        SFID (6top Scheduling Function ID): 0x00
        .... 1101 = SeqNum: 13
        0101 .... = GEN: Lollipop Counter Value (5)
        Metadata: 0x0f00
      FCS: 0x4962 (Correct)
  == Raw Bytes ==

0000 21 ee 9c fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 00 3f 07 a8 c9 00 06 00 5d 00 0f
0020 62 49

```

6top Response to CLEAR 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:01
  Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request, Information Elements Present, Destination Addressing Mode: Long/64-bit, Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1. .... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1. .... = Information Elements Present: True
    .... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
  Sequence Number: 150
  Destination PAN: 0xcafe
  Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
  Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
  Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
      IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE, Length
: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
  Payload IE
    Payload IE, IETF IE, Length: 5
      Payload IE TLV: 0xa805, Type: Payload, Id: IETF IE
        1... .. = Type: Payload (1)
        .010 1... .. = Id: IETF IE (0x5)
        .... .000 0000 0101 = Length: 5
      Sub-ID: 201
      6top IE
        .... 0000 = 6P Version: 0
        ..01 .... = Type: Response (0x1)
        00.. .... = Reserved: 0x0
        Code: 0x00 (SUCCESS)
        SFID (6top Scheduling Function ID): 0x00
        .... 1101 = SeqNum: 13
        0101 .... = GEN: Lollipop Counter Value (5)
      FCS: 0xc52b (Correct)
  == Raw Bytes ==
0000 21 ee 96 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 05 a8 c9 10 00 00 5d 2b c5

```

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This memo only presents example packets exchanged. It does not define any protocol; there are hence no security considerations in this document.

6. Acknowledgments

The authors would like to thank the OpenWSN community, the 6TiSCH working group and the participants at the 6TiSCH plugtests for there feedback which has helped shape this document.

7. References

7.1. Normative References

- [I-D.ietf-6tisch-6top-protocol] Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-07 (work in progress), June 2017.
- [RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <<http://www.rfc-editor.org/info/rfc8025>>.
- [RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <<http://www.rfc-editor.org/info/rfc8138>>.
- [RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <<http://www.rfc-editor.org/info/rfc8180>>.

7.2. External Informative References

- [OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.

Authors' Addresses

Jonathan Munoz (editor)
Gridbee Communications - INRIA
2 rue Simone Iff
Paris 12 75012
France

Email: jonathan.munoz@inria.fr

Emmanuel Riou
Gridbee Communications
ZI Les Bois de Grasse
7 avenue Michel Chevalier
Grasse 06130
France

Email: emmanuel.riou@gridbeecom.com

Dominique Barthel
Orange Labs
28 Chemin du Vieux Chene
Meylan 38240
France

Email: dominique.barthel@orange.com

6TiSCH
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

J. Munoz, Ed.
Inria
D. Barthel
Orange Labs
July 2, 2018

6TiSCH Example Frames
draft-munoz-6tisch-examples-03

Abstract

This draft contains example frames exchanged by nodes implementing 6TiSCH. Both the raw binary and fully parsed contents of each frame is presented. This document can be used as a reference when implementing 6TiSCH.

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 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. TEMPORARY EDITORIAL NOTES	2
2. Tools Used	3
3. Network Topology	3
4. Examples Frames	3
4.1. Enhanced Beacon	4
4.2. Keep-Alive Frame	9
4.3. ACK Frame	10
4.4. Constrained Join Protocol Packets	11
4.5. RPL DIO	21
4.6. RPL DAO	29
4.6.1. RPL DAO from 2	29
4.6.2. RPL DAO from 3	31
4.7. ICMPv6 echo request/reply	36
4.7.1. ping 2	36
4.7.2. ping 3	40
4.8. 6P Commands and Response	48
4.8.1. 6P ADD	48
4.8.2. 6P COUNT	50
4.8.3. 6P DELETE	53
4.8.4. 6P RELOCATE	55
4.8.5. 6P LIST	58
4.8.6. 6P CLEAR	61
5. [TEMPORARY] Known Bugs/Issues	64
6. IANA Considerations	64
7. Security Considerations	64
8. Acknowledgments	64
9. References	64
9.1. Normative References	64
9.2. External Informative References	64
Authors' Addresses	65

1. TEMPORARY EDITORIAL NOTES

This document is an Internet Draft, so work-in-progress by nature. It contains the following work-in-progress elements:

- o "TODO" statements are elements which have not yet been written by the authors for some reason (lack of time, ongoing discussions with no clear consensus, etc). <https://github.com/openwsn-berkeley/openwsn-fw/commit/961c53778fe411533d74ce24918c95400d834199> The statement does indicate that the text will be written at some point.
- o "TEMPORARY" appendices are there to capture current ongoing discussions, or the changelog of the document. These appendices will be removed in the final text.

- o "IANA_*" identifiers are placeholders for numbers assigned by IANA. These placeholders are to be replaced by the actual values they represent after their assignment by IANA.
- o "RFCXXXX" refers to the RFC number of this specification, once published.
- o The string "REMARK" is put before a remark (questions, suggestion, etc) from an author, editor or contributor. These are on-going discussions at the time of writing, and will not be part of the final text.
- o This section will be removed in the final text.

2. Tools Used

All results presented in this document are collected by running the [OpenWSN] firmware in simulation mode and capturing the frame exchanged using Wireshark.

These are the version of the source code used:

1. Wireshark: from version 2.9.0-80-g31aece5d (v2.9.0rc0-80-g31aece5d) and later.
2. OpenWSN firmware: <https://github.com/openwsn-berkeley/openwsn-fw/commit/961c53778fe411533d74ce24918c95400d834199>
3. OpenWSN software: <https://github.com/openwsn-berkeley/openwsn-sw/commit/9c935d15b3e6b7dea5622e6173c04a0a4fd7ae5d>

3. Network Topology

Network prefix: bbbb::/64

MAC address: 14-15-92-cc-00-00-00-0x

```

          PDR=100%          PDR=100%
+-----+          +-----+          +-----+
| x=1 |-----| x=2 |-----| x=3 |
+-----+          +-----+          +-----+
DAGroot

```

4. Examples Frames

4.1. Enhanced Beacon

Enhanced Beacon sent by 1

== Dissected packet ==

```

IEEE 802.15.4 Enhanced Beacon, Dst: Broadcast,
                               Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xea40, Frame Type: Beacon, PAN ID Compression,
Information Elements Present, Destination Addressing Mode: Short/16-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
    .... .000 = Frame Type: Beacon (0x0)
    .... 0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... ..0. = Acknowledge Request: False
    .... .1... = PAN ID Compression: True
    .... ..0. = Sequence Number Suppression: False
    .... ..1. = Information Elements Present: True
    .... 10... = Destination Addressing Mode:
                               Short/16-bit (0x2)
    ..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 196
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
        IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                               Length: 0
        0... .. = Type: Header (0)
        .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
        .... .000 0000 = Length: 0
Payload IEs, MLME IE
    MLME IE
        Payload IE TLV: 0x881a, Type: Payload, Id: MLME IE
            1... .. = Type: Payload (1)
            .000 1... .. = Id: MLME IE (0x1)
            .... .000 0001 1010 = Length: 26
        Time Synchronization IE
            Payload Sub IE (short): 0x1a06, Type: Short, Sub Id (Short):
                               TSCH Synchronization IE
            0... .. = Type: Short (0)
            .001 1010 .... = Sub Id (Short):
                               TSCH Synchronization IE (0x1a)
            .... .0000 0110 = Length: 6
            Absolute Slot Number: 180790

```

```

    Join Metric: 0
    TSCH Timeslot IE
      Payload IE TLV: 0x1c01, Type: Short, Sub Id (Short):
                                TSCH Timeslot IE
      0... .. = Type: Short (0)
      .001 1100 .. = Sub Id (Short):
                                TSCH Timeslot IE (0x1c)
      .... 0000 0001 = Length: 1
    Data: 00
    Channel Hopping IE
      Payload Sub IE (long): 0xc801, Type: Long, Sub Id (Long):
                                Channel Hopping IE
      1... .. = Type: Long (1)
      .100 1... .. = Sub Id (Long): Channel Hopping IE
                                (0x9)
      .... 0000 0001 = Length: 1
    Hopping Sequence ID: 0x00
    TSCH Slotframe and Link IE
      Payload Sub IE (short): 0x1b0a, Type: Short, Sub Id (Short):
                                TSCH Slotframe and Link IE
      0... .. = Type: Short (0)
      .001 1011 .. = Sub Id (Short):
                                TSCH Slotframe and Link IE (0x1b)
      .... 0000 1010 = Length: 10
    Number of Slotframes: 1
    Slotframes [1]
      Slotframe handle: 0
      Slotframe size: 101
      Number of Links: 1
      Link Information
        Timeslot: 0
        Channel Offset: 0
        Link Options: 15
    FCS: 0x75a3 (Correct)

```

== Raw Bytes ==

```

0000  40 ea c4 fe ca ff ff 01 00 00 00 cc 92 15 14 00
0010  3f 1a 88 06 1a 36 c2 02 00 00 00 01 1c 00 01 c8
0020  00 0a 1b 01 00 65 00 01 00 00 00 00 0f a3 75

```

Enhanced Beacon sent by 2

== Dissected packet ==

```

IEEE 802.15.4 Enhanced Beacon, Dst: Broadcast,
                               Src: 14:15:92:cc:00:00:00:02

```

Frame Control Field: 0xea40, Frame Type: Beacon, PAN ID Compression,
Information Elements Present, Destination Addressing Mode: Short/16-bit,
Frame Version: IEEE Std 802.15.4-2015,

Source Addressing Mode: Long/64-bit

```

.....000 = Frame Type: Beacon (0x0)
.....0... = Security Enabled: False
.....0.... = Frame Pending: False
.....0.... = Acknowledge Request: False
.....1... = PAN ID Compression: True
.....0.... = Sequence Number Suppression: False
.....1.... = Information Elements Present: True
.....10... = Destination Addressing Mode:
                                     Short/16-bit (0x2)
..10..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11..... = Source Addressing Mode: Long/64-bit (0x3)

```

Sequence Number: 189

Destination PAN: 0xcacfe

Destination: 0xffff

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Header IEs, Header Termination 1 IE

Header Termination 1 IE (Payload IEs follow)

```

IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                     Length: 0
0... = Type: Header (0)
.011 1111 0... = Id: Header Termination 1 IE (0x7e)
.... .000 0000 = Length: 0

```

Payload IEs, MLME IE

MLME IE

Payload IE TLV: 0x881a, Type: Payload, Id: MLME IE

```

1... = Type: Payload (1)
.000 1... = Id: MLME IE (0x1)
.... .000 0001 1010 = Length: 26

```

Time Synchronization IE

```

Payload Sub IE (short): 0x1a06, Type: Short, Sub Id (Short):
                                     TSCH Synchronization IE
0... = Type: Short (0)
.001 1010 = Sub Id (Short):
                                     TSCH Synchronization IE (0x1a)
.... .0000 0110 = Length: 6

```

Absolute Slot Number: 180790

Join Metric: 1

TSCH Timeslot IE

```

Payload IE TLV: 0x1c01, Type: Short, Sub Id (Short):
                                     TSCH Timeslot IE
0... = Type: Short (0)
.001 1100 = Sub Id (Short):
                                     TSCH Timeslot IE (0x1c)
.... .0000 0001 = Length: 1

```

```

Data: 00
Channel Hopping IE
  Payload Sub IE (long): 0xc801, Type: Long, Sub Id (Long):
                                Channel Hopping IE
      1... .. = Type: Long (1)
      .100 1... .. = Sub Id (Long):
                                Channel Hopping IE (0x9)
        .... .000 0000 0001 = Length: 1
        Hopping Sequence ID: 0x00
TSCH Slotframe and Link IE
  Payload Sub IE (short): 0x1b0a, Type: Short, Sub Id (Short):
                                TSCH Slotframe and Link IE
      0... .. = Type: Short (0)
      .001 1011 .... = Sub Id (Short):
                                TSCH Slotframe and Link IE (0x1b)
        .... .0000 1010 = Length: 10
        Number of Slotframes: 1
        Slotframes [1]
          Slotframe handle: 0
          Slotframe size: 101
          Number of Links: 1
          Link Information
            Timeslot: 0
            Channel Offset: 0
            Link Options: 15
FCS: 0x6ca4 (Correct)

```

== Raw Bytes ==

```

0000  40 ea bd fe ca ff ff 02 00 00 00 cc 92 15 14 00
0010  3f 1a 88 06 1a 36 c2 02 00 00 01 01 1c 00 01 c8
0020  00 0a 1b 01 00 65 00 01 00 00 00 00 0f a4 6c

```

Enhanced Beacon sent by 3

== Dissected packet ==

```

IEEE 802.15.4 Enhanced Beacon, Dst: Broadcast,
                               Src: 14:15:92:cc:00:00:00:03
Frame Control Field: 0xea40, Frame Type: Beacon, PAN ID Compression,
Information Elements Present, Destination Addressing Mode: Short/16-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
  .... .000 = Frame Type: Beacon (0x0)
  .... .0... = Security Enabled: False
  .... .0.... = Frame Pending: False
  .... .0. .... = Acknowledge Request: False
  .... .1... = PAN ID Compression: True

```

```

.....0..... = Sequence Number Suppression: False
.....1..... = Information Elements Present: True
.....10..... = Destination Addressing Mode:
                                     Short/16-bit (0x2)
..10..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11..... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 56
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
Header IEs, Header Termination 1 IE
    Header Termination 1 IE (Payload IEs follow)
        IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                     Length: 0
            0... .. = Type: Header (0)
            .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
            .... ..000 0000 = Length: 0
Payload IEs, MLME IE
    MLME IE
        Payload IE TLV: 0x881a, Type: Payload, Id: MLME IE
            1... .. = Type: Payload (1)
            .000 1... .. = Id: MLME IE (0x1)
            .... .000 0001 1010 = Length: 26
        Time Synchronization IE
            Payload Sub IE (short): 0x1a06, Type: Short, Sub Id (Short):
                                     TSCH Synchronization IE
                0... .. = Type: Short (0)
                .001 1010 .... .. = Sub Id (Short):
                                     TSCH Synchronization IE (0x1a)
                .... ..0000 0110 = Length: 6
            Absolute Slot Number: 180992
            Join Metric: 2
        TSCH Timeslot IE
            Payload IE TLV: 0x1c01, Type: Short, Sub Id (Short):
                                     TSCH Timeslot IE
                0... .. = Type: Short (0)
                .001 1100 .... .. = Sub Id (Short):
                                     TSCH Timeslot IE (0x1c)
                .... ..0000 0001 = Length: 1
            Data: 00
        Channel Hopping IE
            Payload Sub IE (long): 0xc801, Type: Long, Sub Id (Long):
                                     Channel Hopping IE
                1... .. = Type: Long (1)
                .100 1... .. = Sub Id (Long):
                                     Channel Hopping IE (0x9)
                .... .000 0000 0001 = Length: 1
            Hopping Sequence ID: 0x00

```

```

TSCH Slotframe and Link IE
  Payload Sub IE (short): 0x1b0a, Type: Short, Sub Id (Short):
                                TSCH Slotframe and Link IE
                                0... .... = Type: Short (0)
                                .001 1011 .... = Sub Id (Short):
                                    TSCH Slotframe and Link IE (0x1b)
                                .... .... 0000 1010 = Length: 10
  Number of Slotframes: 1
  Slotframes [1]
    Slotframe handle: 0
    Slotframe size: 101
    Number of Links: 1
    Link Information
      Timeslot: 0
      Channel Offset: 0
      Link Options: 15
FCS: 0x045b (Correct)

```

== Raw Bytes ==

```

0000  40 ea 38 fe ca ff ff 03 00 00 00 cc 92 15 14 00
0010  3f 1a 88 06 1a 00 c3 02 00 00 02 01 1c 00 01 c8
0020  00 0a 1b 01 00 65 00 01 00 00 00 00 0f 5b 04

```

4.2. Keep-Alive Frame

Keep Alive 2->1

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02

Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

Long/64-bit

....001 = Frame Type: Data (0x1)
.... 0... = Security Enabled: False
....0 = Frame Pending: False
....1. = Acknowledge Request: True
....0.. = PAN ID Compression: False
.... ...0 = Sequence Number Suppression: False
.... ..0. = Information Elements Present: False
.... 11.. = Destination Addressing Mode: Long/64-bit (0x3)
..10 = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. = Source Addressing Mode: Long/64-bit (0x3)

Sequence Number: 188

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

FCS: 0xba18 (Correct)

== Raw Bytes ==

0000 21 ec bc fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 18 ba

4.3. ACK Frame

ACK Frame

== Dissected packet ==

```

IEEE 802.15.4 Ack, Sequence Number: 57, Dst: 14:15:92:cc:00:00:00:03,
                               Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xee02, Frame Type: Ack,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
    .... .010 = Frame Type: Ack (0x2)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .0... = Acknowledge Request: False
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .1... = Information Elements Present: True
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11... .. = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 57
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Header IEs, Time Correction IE
    Time Correction IE
        IE Header: 0x0f02, Type: Header, Id: Time Correction IE,
                                   Length: 2
            0... .. = Type: Header (0)
            .000 1111 0... .. = Id: Time Correction IE (0x1e)
            .... .000 0010 = Length: 2
        Time Sync Info: 0x0000, Time Correction: 0,
        Nack: Acknowledgement
FCS: 0x4141 (Correct)

```

== Raw Bytes ==

```

0000  02 ee 39 fe ca 03 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 02 0f 00 00 41 41

```

4.4. Constrained Join Protocol Packets

The examples below deviate from [I-D.ietf-6tisch-minimal-security] in that layer 2 security is disabled and the COAP messages are not protected with OSCORE. Therefore, OSCORE COAP option is missing.

Join Request 3->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
Src: 14:15:92:cc:00:00:00:03
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .0... = Information Elements Present: False
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11... .. = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 0
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
FCS: 0xe7da (Correct)
6LoWPAN
    .... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
    100. .... = Routing Header 6lo: Critical Routing Header (0x04)
    ...0 .... = Packet direction (bit O): Up
    .... 0... = Rank-Error (bit R): No
    .... .0... = Forwarding-Error (bit F): No
    .... ..1. = RPL Instance (bit I): Elided
                  (RPL Instance ID: 0)
    .... ...1 = Sender Rank Compression size (bit K): 1 byte
    .... .... 0000 0101 = 6LoRH Type:
                                Routing Protocol Information (0x05)
RPL Instance: 0x00
Sender Rank: 0x15
IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1... = Traffic class and flow label: Version,
                  traffic class, and flow label compressed (0x3)
    .... .0... = Next header: Inline
    .... ..10 = Hop limit: 64 (0x2)
    .... .... 0... = Context identifier extension: False
    .... .... .0... = Source address compression: Stateless
    .... .... ..01 = Source address mode: 64-bits inline (0x0001)
    .... .... .... 0... = Multicast address compression: False
    .... .... .... .0... = Destination address compression: Stateless
    .... .... .... ..01 = Destination address mode:
                                64-bits inline (0x0001)

```

```

[Source context: fe80::]
[Destination context: fe80::]
Next header: UDP (0x11)
Source: fe80::1415:92cc:0:3
Destination: fe80::1415:92cc:0:2
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3,
Dst: fe80::1415:92cc:0:2
  0110 .... = Version: 6
  .... 0000 0000 .... = Traffic Class: 0x00
                        (DSCP: CS0, ECN: Not-ECT)
  .... 0000 00.. .... = Differentiated
                        Services Codepoint: Default (0)
  .... .... ..00 .... = Explicit Congestion
                        Notification: Not ECN-
                        Capable Transport (0)
  .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 38
Next Header: UDP (17)
Hop Limit: 64
Source: fe80::1415:92cc:0:3
Destination: fe80::1415:92cc:0:2
User Datagram Protocol, Src Port: 5683, Dst Port: 5683
Source Port: 5683
Destination Port: 5683
Length: 38
Checksum: 0x7b3e [unverified]
[Checksum Status: Unverified]
[Stream index: 1]
Constrained Application Protocol, Non-Confirmable, POST, MID:47284
  01.. .... = Version: 1
  ..01 .... = Type: Non-Confirmable (1)
  .... 0000 = Token Length: 0
Code: POST (2)
Message ID: 47284
Opt Name: #1: Uri-Host: 6tisch.arpa
  Opt Desc: Type 3, Critical, Unsafe
  0011 .... = Opt Delta: 3
  .... 1011 = Opt Length: 11
  Uri-Host: 6tisch.arpa
Opt Name: #2: Uri-Path: j
  Opt Desc: Type 11, Critical, Unsafe
  1000 .... = Opt Delta: 8
  .... 0001 = Opt Length: 1
  Uri-Path: j
Opt Name: #3: Proxy-Scheme: coap
  Opt Desc: Type 39, Critical, Unsafe
  1101 .... = Opt Delta: 13
  .... 0100 = Opt Length: 4

```

```

    Opt Delta extended: 15
    Proxy-Scheme: coap
End of options marker: 255
[Uri-Path: coap://6tisch.arpa/j]
Payload: Payload Content-Format: application/octet-stream
                                   (no Content-Format), Length: 5
    Payload Desc: application/octet-stream
    [Payload Length: 5]

```

Payload is implicitly encoded as CBOR.
 Decoding from cbor.me is presented below:

```

A1      # map: 1 element
05      # unsigned integer (5): network identifier label
42      # byte string: length 2
CAFE    # "\xCA\xFE": network identifier (PAN ID)

```

== Raw Bytes ==

```

0000  21 ec 00 fe ca 02 00 00 00 cc 92 15 14 03 00 00
0010  00 cc 92 15 14 f1 83 05 15 7a 11 11 14 15 92 cc
0020  00 00 00 03 14 15 92 cc 00 00 00 02 16 33 16 33
0030  00 26 7b 3e 50 02 b8 b4 3b 36 74 69 73 63 68 2e
0040  61 72 70 61 81 6a d4 0f 63 6f 61 70 ff a1 05 42
0050  ca fe da e7

```

Join Request 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..0. .... .... = Information Elements Present: False
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 17
Destination PAN: 0xcafe

```

```

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0x042e (Correct)
6LoWPAN
.... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
100. .... = Routing Header 6lo: Critical Routing Header (0x04)
...0 .... = Packet direction (bit O): Up
.... 0... = Rank-Error (bit R): No
.... .0.. = Forwarding-Error (bit F): No
.... ..1. = RPL Instance (bit I): Elided
                                           (RPL Instance ID: 0)
.... ....1 .... = Sender Rank Compression size (bit K): 1 byte
.... .... 0000 0101 = 6LoRH Type:
                               Routing Protocol Information (0x05)
RPL Instance: 0x00
Sender Rank: 0x0b
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... = Traffic class and flow label: Version, traffic
           class, and flow label compressed (0x3)
.... .0.. = Next header: Inline
.... ..10 = Hop limit: 64 (0x2)
.... .... 0... = Context identifier extension: False
.... .... .1.. = Source address compression: Stateful
.... .... ..01 = Source address mode: 64-bits inline (0x0001)
.... .... .... 0... = Multicast address compression: False
.... .... .... .1.. = Destination address compression: Stateful
.... .... .... ..01 = Destination address mode:
                               64-bits inline (0x0001)
[Source context: bbbb::]
[Destination context: bbbb::]
Next header: UDP (0x11)
Source: bbbb::1415:92cc:0:2
Destination: bbbb::1415:92cc:0:1
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:2,
                               Dst: bbbb::1415:92cc:0:1
0110 .... = Version: 6
.... 0000 0000 .... = Traffic Class: 0x00
                               (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated
                               Services Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
                               Notification: Not ECN-
                               Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 30
Next Header: UDP (17)

```

```

Hop Limit: 64
Source: bbbb::1415:92cc:0:2
Destination: bbbb::1415:92cc:0:1
User Datagram Protocol, Src Port: 5683, Dst Port: 5683
Source Port: 5683
Destination Port: 5683
Length: 30
Checksum: 0x0515 [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
Constrained Application Protocol, Non-Confirmable, POST, MID:47284
01.. .... = Version: 1
..01 .... = Type: Non-Confirmable (1)
.... 0000 = Token Length: 0
Code: POST (2)
Message ID: 47284
Opt Name: #1: Uri-Path: j
    Opt Desc: Type 11, Critical, Unsafe
    1011 .... = Opt Delta: 11
    .... 0001 = Opt Length: 1
    Uri-Path: j
[Expert Info (Warning/Malformed): Invalid Option Number 40]
[Invalid Option Number 40]
[Severity level: Warning]
[Group: Malformed]
Opt Name: #2: Unknown Option: 14 15 92 cc 00 00 00 03
    Opt Desc: Type 40, Elective, Safe
    1101 .... = Opt Delta: 13
    .... 1000 = Opt Length: 8
    Opt Delta extended: 16
    Unknown: 141592cc00000003
End of options marker: 255
[Uri-Path: /j]
Payload: Payload Content-Format: application/octet-stream
        (no Content-Format), Length: 5
    Payload Desc: application/octet-stream
    [Payload Length: 5]

Payload is implicitly encoded as CBOR.
Decoding from cbor.me is presented below:

A1          # map: 1 element
05          # unsigned integer (5): network identifier label
42          # byte string: length 2
CAFE       # "\xCA\xFE": network identifier (PAN ID)

```

== Raw Bytes ==

```

0000  21 ec 11 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 83 05 0b 7a 55 11 14 15 92 cc
0020  00 00 00 02 14 15 92 cc 00 00 00 01 16 33 16 33
0030  00 1e 05 15 50 02 b8 b4 b1 6a d8 10 14 15 92 cc
0040  00 00 00 03 ff a1 05 42 ca fe 2e 04

```

Join Response 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0.... = Frame Pending: False
    .... .1.... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0.... = Sequence Number Suppression: False
    .... .0.... = Information Elements Present: False
    .... 11.... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10.... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11.... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 37
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
FCS: 0x3d41 (Correct)
6LoWPAN
.... 0001 = Page Number: 1
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1.... = Traffic class and flow label:
              Version, traffic class, and flow label
                      compressed (0x3)
.... .0... = Next header: Inline
.... ..10... = Hop limit: 64 (0x2)
.... .... 0... = Context identifier extension: False
.... .... .1... = Source address compression: Stateful
.... .... ..01... = Source address mode: 64-bits inline (0x0001)
.... .... .... 0... = Multicast address compression: False
.... .... .... .1... = Destination address compression: Stateful
.... .... .... ..01 = Destination address mode:
                          64-bits inline (0x0001)
[Source context: bbbb::]
[Destination context: bbbb::]

```

```

Next header: UDP (0x11)
Source: bbbb::1415:92cc:0:1
Destination: bbbb::1415:92cc:0:2
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:1,
Dst: bbbb::1415:92cc:0:2
  0110 .... = Version: 6
  .... 0000 0000 .... = Traffic Class: 0x00
                                (DSCP: CS0, ECN: Not-ECT)
  .... 0000 00.. .... = Differentiated
                                Services Codepoint:
                                Default (0)
  .... .... ..00 .... = Explicit Congestion
                                Notification: Not ECN-
                                Capable Transport (0)
  .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 44
Next Header: UDP (17)
Hop Limit: 64
Source: bbbb::1415:92cc:0:1
Destination: bbbb::1415:92cc:0:2
User Datagram Protocol, Src Port: 5683, Dst Port: 5683
Source Port: 5683
Destination Port: 5683
Length: 44
Checksum: 0x268f [unverified]
[Checksum Status: Unverified]
[Stream index: 2]
Constrained Application Protocol, Non-Confirmable,
2.04 Changed, MID:47284
  01.. .... = Version: 1
  ..01 .... = Type: Non-Confirmable (1)
  .... 0000 = Token Length: 0
Code: 2.04 Changed (68)
Message ID: 47284
[Expert Info (Warning/Malformed): Invalid Option Number 40]
  [Invalid Option Number 40]
  [Severity level: Warning]
  [Group: Malformed]
Opt Name: #1: Unknown Option: 14 15 92 cc 00 00 00 03
Opt Desc: Type 40, Elective, Safe
  1101 .... = Opt Delta: 13
  .... 1000 = Opt Length: 8
Opt Delta extended: 27
Unknown: 141592cc000000003
End of options marker: 255
Payload: Payload Content-Format: application/octet-stream
                                (no Content-Format), Length: 2
Payload Desc: application/octet-stream

```


[Payload Length: 21]

Payload is implicitly encoded as CBOR.
Decoding from cbor.me is presented below:

```
A1      # map: 1 element
02      # unsigned integer (2): link-layer key set label
82      # array: 2 elements
01      # unsigned integer (1): key_index value
50      # byte string: length 16
11111111111111111111111111111111 # key value
```

== Raw Bytes ==

```
0000    21 ec 25 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010    00 cc 92 15 14 f1 7a 55 11 14 15 92 cc 00 00 00
0020    01 14 15 92 cc 00 00 00 02 16 33 16 33 00 2c 26
0030    8f 50 44 b8 b4 d8 1b 14 15 92 cc 00 00 00 03 ff
0040    a1 02 82 01 50 11 11 11 11 11 11 11 11 11 11
0050    11 11 11 11 11 41 3d
```

Join Response 2->3

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:03,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
.... .... .001 = Frame Type: Data (0x1)
.... .... .0... = Security Enabled: False
.... .... .0 .... = Frame Pending: False
.... .... .1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... .... 0 .... = Sequence Number Suppression: False
.... ..0. .... = Information Elements Present: False
.... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 19
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0x9e69 (Correct)
6LoWPAN
.... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
```

```

100. .... = Routing Header 6lo: Critical Routing Header (0x04)
...0 .... = Packet direction (bit O): Up
.... 0... = Rank-Error (bit R): No
.... .0.. = Forwarding-Error (bit F): No
.... ..1. = RPL Instance (bit I): Elided
                                           (RPL Instance ID: 0)
.... ....1 .... = Sender Rank Compression size (bit K): 1 byte
.... .... 0000 0101 = 6loRH Type: Routing Protocol Information
                                           (0x05)

RPL Instance: 0x00
Sender Rank: 0x0b
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... = Traffic class and flow label: Version,
~         traffic class, and flow label compressed (0x3)
.... .0.. = Next header: Inline
.... ..10 = Hop limit: 64 (0x2)
.... .... 0... = Context identifier extension: False
.... .... .0.. = Source address compression: Stateless
.... .... ..01 = Source address mode: 64-bits inline (0x0001)
.... .... .... 0... = Multicast address compression: False
.... .... .... .0.. = Destination address compression: Stateless
.... .... .... ..01 = Destination address mode:
                                           64-bits inline (0x0001)

[Source context: fe80::]
[Destination context: fe80::]
Next header: UDP (0x11)
Source: fe80::1415:92cc:0:2
Destination: fe80::1415:92cc:0:3
Internet Protocol Version 6, Src: fe80::1415:92cc:0:2,
Dst: fe80::1415:92cc:0:3
  0110 .... = Version: 6
  .... 0000 0000 .... = Traffic Class:
                        0x00 (DSCP: CS0, ECN: Not-ECT)
  .... 0000 00.. .... = Differentiated
                        Services Codepoint: Default (0)
  .... .... ..00 .... = Explicit Congestion
                        Notification: Not ECN-
                        Capable Transport (0)
  .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 34
Next Header: UDP (17)
Hop Limit: 64
Source: fe80::1415:92cc:0:2
Destination: fe80::1415:92cc:0:3
User Datagram Protocol, Src Port: 5683, Dst Port: 5683
Source Port: 5683
Destination Port: 5683

```

```

Length: 34
Checksum: 0x364a [unverified]
[Checksum Status: Unverified]
[Stream index: 1]
Constrained Application Protocol, Non-Confirmable, 2.04 Changed,
MID:47284
01.. .... = Version: 1
..01 .... = Type: Non-Confirmable (1)
.... 0000 = Token Length: 0
Code: 2.04 Changed (68)
Message ID: 47284
End of options marker: 255
Payload: Payload Content-Format: application/octet-stream
(no Content-Format), Length: 2
Payload Desc: application/octet-stream
[Payload Length: 21]

```

Payload is implicitly encoded as CBOR.
 Decoding from cbor.me is presented below:

```

A1      # map: 1 element
02      # unsigned integer (2): link-layer key set label
82      # array: 2 elements
01      # unsigned integer (1): key_index value
50      # byte string: length 16
1111111111111111111111111111111111 # key value

```

== Raw Bytes ==

```

0000  21 ec 13 fe ca 03 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 83 05 0b 7a 11 11 14 15 92 cc
0020  00 00 00 02 14 15 92 cc 00 00 00 03 16 33 16 33
0030  00 22 36 4a 50 44 b8 b4 ff a1 02 82 01 50 11 11
0040  11 11 11 11 11 11 11 11 11 11 11 11 11 11 69 9e

```

4.5. RPL DIO

RPL DIO sent by 1

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xe841, Frame Type: Data, PAN ID Compression,
Destination Addressing Mode: Short/16-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
.... .... .... .001 = Frame Type: Data (0x1)

```

```

..... 0... = Security Enabled: False
..... 0... = Frame Pending: False
..... 0... = Acknowledge Request: False
..... 1... = PAN ID Compression: True
..... 0... = Sequence Number Suppression: False
..... 0... = Information Elements Present: False
..... 10... = Destination Addressing Mode:
                                     Short/16-bit (0x2)
..10 ..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11... .. = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 197
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
FCS: 0xeb21 (Correct)
6LoWPAN
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... .. = Traffic class and flow label: Version,
               traffic class, and flow label compressed (0x3)
..... 0... .. = Next header: Inline
..... 10... .. = Hop limit: 64 (0x2)
..... 0... .. = Context identifier extension: False
..... 0... .. = Source address compression: Stateless
..... 11... .. = Source address mode: Compressed (0x0003)
..... 1... .. = Multicast address compression: True
..... 0... .. = Destination address compression: Stateless
..... 11... .. = Destination address mode:
                                     8-bits inline (0x0003)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1615:92cc:0:1
Destination: ff02::1a
Internet Protocol Version 6, Src: fe80::1615:92cc:0:1, Dst: ff02::1a
0110 .... = Version: 6
.... 0000 0000 ..... = Traffic Class: 0x00
                           (DSCP: CS0, ECN: Not-ECT)
.... 0000 00... .. = Differentiated Services
                           Codepoint: Default (0)
.... 00... .. = Explicit Congestion
                           Notification:
                           Not ECN-Capable Transport (0)
.... 0000 0000 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 76
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: fe80::1615:92cc:0:1

```

```

Destination: ff02::1a
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 1 (DODAG Information Object)
Checksum: 0xbccd [correct]
[Checksum Status: Good]
RPLInstanceID: 0
Version: 0
Rank: 256
Flags: 0x88, Grounded (G), Mode of Operation (MOP): Non-Storing
                                           Mode of Operation
    1... .... = Grounded (G): True
    .0.. .... = Zero: False
    ..00 1... = Mode of Operation (MOP): Non-Storing
                                           Mode of Operation (0x1)
    .... .000 = DODAG Preference: 0
Destination Advertisement Trigger Sequence Number (DTSN): 51
Flags: 0x00
Reserved: 00
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Prefix Information bbbb::/64)
  Type: Prefix Information (8)
  Length: 30
  Prefix Length: 64
  Flag: 0x60, Auto Address Config, Router Address
    0... .... = On Link: Not set
    .1.. .... = Auto Address Config: Set
    ..1. .... = Router Address: Set
    ...0 0000 = Reserved: 0
  Valid Lifetime: Infinity (4294967295)
  Preferred Lifetime: Infinity (4294967295)
  Reserved
  Destination Prefix: bbbb::
ICMPv6 RPL Option (DODAG configuration)
  Type: DODAG configuration (4)
  Length: 14
  Flag: 0x00
    0000 .... = Reserved: 0
    .... 0... = Authentication Enabled: Not set
    .... .000 = Path Control Size: 0
  DIOIntervalDoublings: 8
  DIOIntervalMin: 12
  DIORedundancyConstant: 0
  MaxRankInc: 8
  MinHopRankInc: 1
  OCP (Objective Code Point): 0

```

Reserved: 0
 Default Lifetime: 255
 Lifetime Unit: 65535

== Raw Bytes ==

```
0000  41 e8 c5 fe ca ff ff 01 00 00 00 cc 92 15 14 7a
0010  3b 3a 1a 9b 01 bc cd 00 00 01 00 88 33 00 00 bb
0020  bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01 08
0030  1e 40 60 ff ff ff ff ff ff ff ff 00 00 00 00 bb
0040  bb 00 00 00 00 00 00 00 00 00 00 00 00 00 04
0050  0e 00 08 0c 00 00 08 00 01 00 00 00 ff ff ff 21
0060  eb
```

RPL DIO sent by 2

== Dissected packet ==

```
IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xe841, Frame Type: Data, PAN ID Compression,
Destination Addressing Mode: Short/16-bit,
Frame Version: IEEE Std 802.15.4-2015,
Source Addressing Mode: Long/64-bit
  .... .... .001 = Frame Type: Data (0x1)
  .... .... 0... = Security Enabled: False
  .... .... .0... = Frame Pending: False
  .... .... .0. .... = Acknowledge Request: False
  .... .... .1.. .... = PAN ID Compression: True
  .... .... .0 .... = Sequence Number Suppression: False
  .... .... .0. .... = Information Elements Present: False
  .... 10.. .... = Destination Addressing Mode:
                                Short/16-bit (0x2)
  ..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
  11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 197
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0xab62 (Correct)
6LoWPAN
IPHC Header
  011. .... = Pattern: IP header compression (0x03)
  ...1 1... .... = Traffic class and flow label: Version,
                  traffic class, and flow label compressed (0x3)
  .... .0.. .... = Next header: Inline
  .... ..10 .... = Hop limit: 64 (0x2)
  .... .... 0... .... = Context identifier extension: False
```

```

..... .0.. .... = Source address compression: Stateless
..... ..11 .... = Source address mode: Compressed (0x0003)
..... .... 1... = Multicast address compression: True
..... .... .0.. = Destination address compression: Stateless
..... .... ..11 = Destination address mode:
                                     8-bits inline (0x0003)

[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1615:92cc:0:2
Destination: ff02::1a
Internet Protocol Version 6, Src: fe80::1615:92cc:0:2, Dst: ff02::1a
0110 .... = Version: 6
..... 0000 0000 .... = Traffic Class:
                                     0x00 (DSCP: CS0, ECN: Not-ECT)
..... 0000 00.. .... = Differentiated Services
                                     Codepoint: Default (0)
..... .... ..00 .... = Explicit Congestion
                                     Notification:
                                     Not ECN-Capable Transport (0)
..... .... 0000 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 76
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: fe80::1615:92cc:0:2
Destination: ff02::1a
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 1 (DODAG Information Object)
Checksum: 0xbbcc [correct]
[Checksum Status: Good]
RPLInstanceID: 0
Version: 0
Rank: 512
Flags: 0x88, Grounded (G), Mode of Operation (MOP): Non-Storing
                                     Mode of Operation
1... .... = Grounded (G): True
.0.. .... = Zero: False
..00 1... = Mode of Operation (MOP): Non-Storing
                                     Mode of Operation (0x1)
..... .000 = DODAG Preference: 0
Destination Advertisement Trigger Sequence Number (DTSN): 51
Flags: 0x00
Reserved: 00
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Prefix Information bbbb::/64)

```

```

Type: Prefix Information (8)
Length: 30
Prefix Length: 64
Flag: 0x60, Auto Address Config, Router Address
    0... .. = On Link: Not set
    .1... .. = Auto Address Config: Set
    ..1. .... = Router Address: Set
    ...0 0000 = Reserved: 0
Valid Lifetime: Infinity (4294967295)
Preferred Lifetime: Infinity (4294967295)
Reserved
Destination Prefix: bbbb::
ICMPv6 RPL Option (DODAG configuration)
Type: DODAG configuration (4)
Length: 14
Flag: 0x00
    0000 .... = Reserved: 0
    .... 0... = Authentication Enabled: Not set
    .... .000 = Path Control Size: 0
DIOIntervalDoublings: 8
DIOIntervalMin: 12
DIORedundancyConstant: 0
MaxRankInc: 8
MinHopRankInc: 1
OCP (Objective Code Point): 0
Reserved: 0
Default Lifetime: 255
Lifetime Unit: 65535

```

== Raw Bytes ==

```

0000  41 e8 c5 fe ca ff ff 02 00 00 00 cc 92 15 14 7a
0010  3b 3a 1a 9b 01 bb cc 00 00 02 00 88 33 00 00 bb
0020  bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01 08
0030  1e 40 60 ff ff ff ff ff ff ff 00 00 00 00 bb
0040  bb 00 00 00 00 00 00 00 00 00 00 00 00 00 04
0050  0e 00 08 0c 00 00 08 00 01 00 00 00 ff ff ff 62
0060  ab

```

RPL DIO sent by 3

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:03
Frame Control Field: 0xe841, Frame Type: Data, PAN ID Compression,
Destination Addressing Mode: Short/16-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

```



```

................................................................001 = Frame Type: Data (0x1)
................................................................0... = Security Enabled: False
................................................................0.... = Frame Pending: False
................................................................0. .... = Acknowledge Request: False
................................................................1... = PAN ID Compression: True
.....0 ..... = Sequence Number Suppression: False
.....0. .... = Information Elements Present: False
.....10... .. = Destination Addressing Mode:
                                                    Short/16-bit (0x2)
..10 ..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11... .. = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 66
Destination PAN: 0xcafe
Destination: 0xffff
Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
FCS: 0x7daa (Correct)
6LoWPAN
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1... .. = Traffic class and flow label: Version,
               traffic class, and flow label compressed (0x3)
.....0... .. = Next header: Inline
.....10 ..... = Hop limit: 64 (0x2)
.....0... .. = Context identifier extension: False
.....0... .. = Source address compression: Stateless
.....11 ..... = Source address mode: Compressed (0x0003)
.....1... .. = Multicast address compression: True
.....0... .. = Destination address compression: Stateless
.....11 ..... = Destination address mode:
                                                    8-bits inline (0x0003)
[Source context: fe80::]
[Destination context: fe80::]
Next header: ICMPv6 (0x3a)
Source: fe80::1615:92cc:0:3
Destination: ff02::1a
Internet Protocol Version 6, Src: fe80::1615:92cc:0:3, Dst: ff02::1a
0110 .... = Version: 6
.... 0000 0000 ..... = Traffic Class:
                        0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. ..... = Differentiated Services
                        Codepoint: Default (0)
.... ....00 ..... = Explicit Congestion
                        Notification:
                        Not ECN-Capable Transport (0)
.... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 76
Next Header: ICMPv6 (58)

```

```

Hop Limit: 64
Source: fe80::1615:92cc:0:3
Destination: ff02::1a
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 1 (DODAG Information Object)
Checksum: 0xbabe [correct]
[Checksum Status: Good]
RPLInstanceID: 0
Version: 0
Rank: 781
Flags: 0x88, Grounded (G), Mode of Operation (MOP): Non-Storing
                                           Mode of Operation
    1... .... = Grounded (G): True
    .0.. .... = Zero: False
    ..00 1... = Mode of Operation (MOP): Non-Storing
                                           Mode of Operation (0x1)
    .... .000 = DODAG Preference: 0
Destination Advertisement Trigger Sequence Number (DTSN): 51
Flags: 0x00
Reserved: 00
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Prefix Information bbbb::/64)
    Type: Prefix Information (8)
    Length: 30
    Prefix Length: 64
    Flag: 0x60, Auto Address Config, Router Address
        0... .... = On Link: Not set
        .1.. .... = Auto Address Config: Set
        ..1. .... = Router Address: Set
        ...0 0000 = Reserved: 0
    Valid Lifetime: Infinity (4294967295)
    Preferred Lifetime: Infinity (4294967295)
    Reserved
    Destination Prefix: bbbb::
ICMPv6 RPL Option (DODAG configuration)
    Type: DODAG configuration (4)
    Length: 14
    Flag: 0x00
        0000 .... = Reserved: 0
        .... 0... = Authentication Enabled: Not set
        .... .000 = Path Control Size: 0
    DIOIntervalDoublings: 8
    DIOIntervalMin: 12
    DIORedundancyConstant: 0
    MaxRankInc: 8

```

```

MinHopRankInc: 1
OCP (Objective Code Point): 0
Reserved: 0
Default Lifetime: 255
Lifetime Unit: 65535

```

== Raw Bytes ==

```

0000  41 e8 42 fe ca ff ff 03 00 00 00 cc 92 15 14 7a
0010  3b 3a 1a 9b 01 ba be 00 00 03 0d 88 33 00 00 bb
0020  bb 00 00 00 00 00 00 14 15 92 cc 00 00 01 08
0030  1e 40 60 ff ff ff ff ff ff ff ff 00 00 00 00 bb
0040  bb 00 00 00 00 00 00 00 00 00 00 00 00 00 04
0050  0e 00 08 0c 00 00 08 00 01 00 00 00 ff ff ff aa
0060  7d

```

4.6. RPL DAO

4.6.1. RPL DAO from 2

[RPL DAO from 2] 2->1

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                    Long/64-bit
..... .001 = Frame Type: Data (0x1)
..... 0... = Security Enabled: False
..... 0... = Frame Pending: False
..... 1... = Acknowledge Request: True
..... 0... = PAN ID Compression: False
..... 0... = Sequence Number Suppression: False
..... 0... = Information Elements Present: False
..... 11.. = Destination Addressing Mode: Long/64-bit (0x3)
..10 ..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. ..... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 223
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0xc883 (Correct)

```

6LoWPAN

.... 0001 = Page Number: 1

6LoRH: Routing Protocol Information

100. = Routing Header 6lo: Critical Routing Header (0x04)

...0 = Packet direction (bit O): Up

.... 0... = Rank-Error (bit R): No

.... .0.. = Forwarding-Error (bit F): No

.... ..1. = RPL Instance (bit I): Elided

(RPL Instance ID: 0)

....1 = Sender Rank Compression size (bit K): 1 byte

.... 0000 0101 = 6LoRH Type: Routing Protocol Information

(0x05)

RPL Instance: 0x00

Sender Rank: 0x02

IPHC Header

011. = Pattern: IP header compression (0x03)

...1 1... = Traffic class and flow label: Version,
traffic class, and flow label compressed (0x3)

.... .0.. = Next header: Inline

.... ..10 = Hop limit: 64 (0x2)

.... 0... = Context identifier extension: False

....1.. = Source address compression: Stateful

....01 = Source address mode: 64-bits inline (0x0001)

.... 0... = Multicast address compression: False

....1.. = Destination address compression: Stateful

....01 = Destination address mode:

64-bits inline (0x0001)

[Source context: bbbb::]

[Destination context: bbbb::]

Next header: ICMPv6 (0x3a)

Source: bbbb::1415:92cc:0:2

Destination: bbbb::1415:92cc:0:1

Internet Protocol Version 6, Src: bbbb::1415:92cc:0:2,
Dst: bbbb::1415:92cc:0:1

0110 = Version: 6

.... 0000 0000 = Traffic Class:
0x00 (DSCP: CS0, ECN: Not-ECT).... 0000 00.. = Differentiated Services
Codepoint: Default (0)....00 = Explicit Congestion
Notification:

Not ECN-Capable Transport (0)

.... 0000 0000 0000 0000 0000 = Flow Label: 0x00000

Payload Length: 66

Next Header: ICMPv6 (58)

Hop Limit: 64

Source: bbbb::1415:92cc:0:2

Destination: bbbb::1415:92cc:0:1

```

[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 2 (Destination Advertisement Object)
Checksum: 0x3aa5 [correct]
[Checksum Status: Good]
RPLInstanceID: 0
Flags: 0x40, DODAGID Present (D)
    0... .... = DAO-ACK Request (K): False
    .1... .... = DODAGID Present (D): True
    ..00 0000 = Reserved: 0
Reserved: 00
DAO Sequence: 49
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (RPL Target bbbb::1415:92cc:0:3/128)
    Type: RPL Target (5)
    Length: 18
    Reserved
    Target Length: 128
    Target: bbbb::1415:92cc:0:3
ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:1)
    Type: Transit Information (6)
    Length: 20
    Flags: 0x00
        0... .... = External: Not set
        .000 0000 = Reserved: 0
    Path Control: 0
    Path Sequence: 48
    Path Lifetime: 170
    Parent Address: bbbb::1415:92cc:0:1

```

== Raw Bytes ==

```

0000  21 ec df fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 83 05 02 7a 55 3a 14 15 92 cc
0020  00 00 00 02 14 15 92 cc 00 00 00 01 9b 02 3a a5
0030  00 40 00 31 bb bb 00 00 00 00 00 00 14 15 92 cc
0040  00 00 00 01 05 12 00 80 bb bb 00 00 00 00 00 00
0050  14 15 92 cc 00 00 00 03 06 14 00 00 30 aa bb bb
0060  00 00 00 00 00 00 14 15 92 cc 00 00 00 01 83 c8

```

4.6.2. RPL DAO from 3

[RPL DAO from 3] 3->2

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
 Src: 14:15:92:cc:00:00:00:03
 Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
 Destination Addressing Mode: Long/64-bit,
 Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
 Long/64-bit

```

..... .001 = Frame Type: Data (0x1)
..... 0... = Security Enabled: False
..... .0... = Frame Pending: False
..... .1... = Acknowledge Request: True
..... .0... = PAN ID Compression: False
..... .0... = Sequence Number Suppression: False
..... .0... = Information Elements Present: False
..... 11... = Destination Addressing Mode: Long/64-bit (0x3)
..10... = Frame Version: IEEE Std 802.15.4-2015 (2)
11... = Source Addressing Mode: Long/64-bit (0x3)

```

Sequence Number: 6

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)

FCS: 0xee92 (Correct)

6LoWPAN

```

..... 0001 = Page Number: 1

```

6LoRH: Routing Protocol Information

```

100. .... = Routing Header 6lo: Critical Routing Header (0x04)
...0 .... = Packet direction (bit O): Up
.... 0... = Rank-Error (bit R): No
.... .0... = Forwarding-Error (bit F): No
.... .1... = RPL Instance (bit I):
                               Elided (RPL Instance ID: 0)
.... ...0 .... = Sender Rank Compression size (bit K):
                               2 bytes
.... .... 0000 0101 = 6LoRH Type:
                               Routing Protocol Information (0x05)

```

RPL Instance: 0x00

Sender Rank: 0x0c2b

IPHC Header

```

011. .... = Pattern: IP header compression (0x03)
...1 1... = Traffic class and flow label: Version,
           traffic class, and flow label compressed (0x3)
.... .0... = Next header: Inline
.... .10... = Hop limit: 64 (0x2)
.... .... 0... = Context identifier extension: False
.... .... .1... = Source address compression: Stateful
.... .... .01... = Source address mode:
                               64-bits inline (0x0001)
.... .... 0... = Multicast address compression: False
.... .... .1... = Destination address compression: Stateful

```

```

.....01 = Destination address mode:
                                     64-bits inline (0x0001)
[Source context: bbbb::]
[Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Source: bbbb::1415:92cc:0:3
Destination: bbbb::1415:92cc:0:1
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:3,
                                Dst: bbbb::1415:92cc:0:1
0110 .... = Version: 6
.... 0000 0000 .... = Traffic Class:
                                0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated
                                Services Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
                                Notification:
                                Not ECN-Capable Transport (0)
.... .... 0000 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 46
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: bbbb::1415:92cc:0:3
Destination: bbbb::1415:92cc:0:1
Internet Control Message Protocol v6
Type: RPL Control (155)
Code: 2 (Destination Advertisement Object)
Checksum: 0xd218 [correct]
[Checksum Status: Good]
RPLInstanceID: 0
Flags: 0x40, DODAGID Present (D)
  0... .... = DAO-ACK Request (K): False
  .1.. .... = DODAGID Present (D): True
  ..00 0000 = Reserved: 0
Reserved: 00
DAO Sequence: 2
DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:2)
  Type: Transit Information (6)
  Length: 20
  Flags: 0x00
    0... .... = External: Not set
    .000 0000 = Reserved: 0
  Path Control: 0
  Path Sequence: 1
  Path Lifetime: 170
  Parent Address: bbbb::1415:92cc:0:2

== Raw Bytes ==

```

```

0000  21 ec 06 fe ca 02 00 00 00 cc 92 15 14 03 00 00
0010  00 cc 92 15 14 f1 82 05 0c 2b 7a 55 3a 14 15 92
0020  cc 00 00 00 03 14 15 92 cc 00 00 00 01 9b 02 d2
0030  18 00 40 00 02 bb bb 00 00 00 00 00 00 14 15 92
0040  cc 00 00 00 01 06 14 00 00 01 aa bb bb 00 00 00
0050  00 00 00 14 15 92 cc 00 00 00 02 92 ee

```

[RPL DAO from 3] 2->1

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,

Src: 14:15:92:cc:00:00:00:02

Frame Control Field: 0xec21, Frame Type: Data,

Acknowledge Request, Destination Addressing Mode: Long/64-bit,

Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

Long/64-bit

```

.... .... .001 = Frame Type: Data (0x1)
.... .... .0... = Security Enabled: False
.... .... .0.... = Frame Pending: False
.... .... .1.... = Acknowledge Request: True
.... .... .0... = PAN ID Compression: False
.... .... .0.... = Sequence Number Suppression: False
.... ..0.... = Information Elements Present: False
.... 11.... = Destination Addressing Mode: Long/64-bit (0x3)
..10.... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.... = Source Addressing Mode: Long/64-bit (0x3)

```

Sequence Number: 161

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

FCS: 0x4f42 (Correct)

6LoWPAN

.... 0001 = Page Number: 1

6LoRH: Routing Protocol Information

100. = Routing Header 6Lo: Critical Routing Header (0x04)

...0 = Packet direction (bit O): Up

.... 0... = Rank-Error (bit R): No

.... .0... = Forwarding-Error (bit F): No

.... ..1. = RPL Instance (bit I):

Elided (RPL Instance ID: 0)

.... ...0 = Sender Rank Compression size (bit K):

2 bytes

.... 0000 0101 = 6LoRH Type:

Routing Protocol Information (0x05)

RPL Instance: 0x00

Sender Rank: 0x0229


```

IPHC Header
  011. .... = Pattern: IP header compression (0x03)
  ...1 1... .... = Traffic class and flow label: Version,
                                traffic class, and flow label
                                compressed (0x3)
  .... .0... .... = Next header: Inline
  .... ..10 .... = Hop limit: 64 (0x2)
  .... .... 0... .... = Context identifier extension: False
  .... .... .1.. .... = Source address compression: Stateful
  .... .... ..01 .... = Source address mode:
                                64-bits inline (0x0001)
  .... .... .... 0... = Multicast address compression: False
  .... .... .... .1.. = Destination address compression: Stateful
  .... .... .... ..01 = Destination address mode:
                                64-bits inline (0x0001)
  [Source context: bbbb::]
  [Destination context: bbbb::]
  Next header: ICMPv6 (0x3a)
  Source: bbbb::1415:92cc:0:3
  Destination: bbbb::1415:92cc:0:1
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:3,
                                Dst: bbbb::1415:92cc:0:1
  0110 .... = Version: 6
  .... 0000 0000 .... = Traffic Class:
                                0x00 (DSCP: CS0, ECN: Not-ECT)
  .... 0000 00.. .... = Differentiated
                                Services Codepoint: Default (0)
  .... .... ..00 .... = Explicit Congestion
                                Notification:
                                Not ECN-Capable Transport (0)
  .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
  Payload Length: 46
  Next Header: ICMPv6 (58)
  Hop Limit: 64
  Source: bbbb::1415:92cc:0:3
  Destination: bbbb::1415:92cc:0:1
Internet Control Message Protocol v6
  Type: RPL Control (155)
  Code: 2 (Destination Advertisement Object)
  Checksum: 0xd218 [correct]
  [Checksum Status: Good]
  RPLInstanceID: 0
  Flags: 0x40, DODAGID Present (D)
    0... .... = DAO-ACK Request (K): False
    .1.. .... = DODAGID Present (D): True
    ..00 0000 = Reserved: 0
  Reserved: 00
  DAO Sequence: 2

```

```

DODAGID: bbbb::1415:92cc:0:1
ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:2)
  Type: Transit Information (6)
  Length: 20
  Flags: 0x00
    0... .... = External: Not set
    .000 0000 = Reserved: 0
  Path Control: 0
  Path Sequence: 1
  Path Lifetime: 170
  Parent Address: bbbb::1415:92cc:0:2

```

== Raw Bytes ==

```

0000  21 ec a1 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 82 05 02 29 7a 55 3a 14 15 92
0020  cc 00 00 00 03 14 15 92 cc 00 00 00 01 9b 02 d2
0030  18 00 40 00 02 bb bb 00 00 00 00 00 00 14 15 92
0040  cc 00 00 00 01 06 14 00 00 01 aa bb bb 00 00 00
0050  00 00 00 14 15 92 cc 00 00 00 02 42 4f

```

4.7. ICMPv6 echo request/reply

4.7.1. ping 2

[ping 2] ICMPv6 echo request 1->2

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                      Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                    Long/64-bit
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..0. .... .... = Information Elements Present: False
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 74
Destination PAN: 0xcafe

```

```

Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
FCS: 0x6ec7 (Correct)
6LoWPAN
.... 0001 = Page Number: 1
IPHC Header
  011. .... = Pattern: IP header compression (0x03)
  ...1 1... .... = Traffic class and flow label: Version,
                  traffic class, and flow label compressed (0x3)
  .... .0.. .... = Next header: Inline
  .... ..00 .... = Hop limit: Inline (0x0)
  .... .... 0... .... = Context identifier extension: False
  .... .... .1.. .... = Source address compression: Stateful
  .... .... ..01 .... = Source address mode: 64-bits inline (0x0001)
  .... .... .... 0... = Multicast address compression: False
  .... .... .... .1.. = Destination address compression: Stateful
  .... .... .... ..01 = Destination address mode:
                          64-bits inline (0x0001)

  [Source context: bbbb::]
  [Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Hop limit: 128
Source: bbbb::1
Destination: bbbb::1415:92cc:0:2
Internet Protocol Version 6, Src: bbbb::1, Dst: bbbb::1415:92cc:0:2
0110 .... = Version: 6
.... 0000 0000 .... = Traffic Class:
                          0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated Services
                          Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
                          Notification:
                          Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 40
Next Header: ICMPv6 (58)
Hop Limit: 128
Source: bbbb::1
Destination: bbbb::1415:92cc:0:2
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0xb662 [correct]
[Checksum Status: Good]
Identifier: 0x0001
Sequence: 58

```

[Response In: 2369]

Data (32 bytes)

Data: 6162636465666768696a6b6c6d6e6f707172737475767761...

[Length: 32]

== Raw Bytes ==

```
0000  21 ec 4a fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 f1 78 55 3a 80 00 00 00 00 00 00
0020  00 01 14 15 92 cc 00 00 00 02 80 00 b6 62 00 01
0030  00 3a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e
0040  6f 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67
0050  68 69 c7 6e
```

[ping 2] ICMPv6 echo reply 2->1

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,

Src: 14:15:92:cc:00:00:00:02

Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,

Destination Addressing Mode: Long/64-bit,

Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

Long/64-bit

```
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..0. .... .... = Information Elements Present: False
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
```

Sequence Number: 6

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

FCS: 0x1763 (Correct)

6LoWPAN

.... 0001 = Page Number: 1

6LoRH: Routing Protocol Information

100. = Routing Header 6lo: Critical Routing Header (0x04)

...0 = Packet direction (bit O): Up

.... 0... = Rank-Error (bit R): No

.... .0.. = Forwarding-Error (bit F): No

.... ..1. = RPL Instance (bit I):

Elided (RPL Instance ID: 0)

```

.....0..... = Sender Rank Compression size (bit K): 2 bytes
..... 0000 0101 = 6loRH Type:
                                Routing Protocol Information (0x05)
RPL Instance: 0x00
Sender Rank: 0x028a
IPHC Header
011. .... = Pattern: IP header compression (0x03)
...1 1.... = Traffic class and flow label: Version,
            traffic class, and flow label compressed (0x3)
.....0... = Next header: Inline
.....10... = Hop limit: 64 (0x2)
..... 0... = Context identifier extension: False
..... 1... = Source address compression: Stateful
..... 01... = Source address mode: 64-bits inline (0x0001)
..... 0... = Multicast address compression: False
..... 1... = Destination address compression: Stateful
..... 01... = Destination address mode:
                                64-bits inline (0x0001)
[Source context: bbbb::]
[Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Source: bbbb::1415:92cc:0:2
Destination: bbbb::1
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:2, Dst: bbbb::1
0110 .... = Version: 6
..... 0000 0000 ..... = Traffic Class:
                                0x00 (DSCP: CS0, ECN: Not-ECT)
..... 0000 00.. ..... = Differentiated Services
                                Codepoint: Default (0)
..... ..00 ..... = Explicit Congestion
                                Notification:
                                Not ECN-Capable Transport (0)
..... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 40
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: bbbb::1415:92cc:0:2
Destination: bbbb::1
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Echo (ping) reply (129)
Code: 0
Checksum: 0xb562 [correct]
[Checksum Status: Good]
Identifier: 0x0001
Sequence: 58
[Response To: 2366]

```

[Response Time: 1857.163 ms]

Data (32 bytes)

Data: 6162636465666768696a6b6c6d6e6f707172737475767761...

[Length: 32]

== Raw Bytes ==

```
0000  21 ec 06 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 f1 82 05 02 8a 7a 55 3a 14 15 92
0020  cc 00 00 00 02 00 00 00 00 00 00 01 81 00 b5
0030  62 00 01 00 3a 61 62 63 64 65 66 67 68 69 6a 6b
0040  6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64
0050  65 66 67 68 69 63 17
```

4.7.2. ping 3

[ping 3] ICMPv6 echo request 1->2

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,

Src: 14:15:92:cc:00:00:00:01

Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,

Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

Long/64-bit

```
.... .... .001 = Frame Type: Data (0x1)
.... .... 0... = Security Enabled: False
.... .... .0... = Frame Pending: False
.... .... .1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..0. .... .... = Information Elements Present: False
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
```

Sequence Number: 163

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

FCS: 0xd31e (Correct)

6LoWPAN

.... 0001 = Page Number: 1

6LoRH: Routing Header 3, 8 byte compression

100. = Routing Header 6lo: Critical Routing Header (0x04)

...0 0000 = 6LoRH Hop Number-1: 0x00

.... 0000 0011 = 6LoRH Type: Routing Header 3,

```

Source/8, Delta: ::1415:92cc:0:2
IPHC Header
  011. .... = Pattern: IP header compression (0x03)
  ....1 1... .... = Traffic class and flow label: Version,
                    traffic class, and flow label compressed (0x3)
  .... .0.. .... = Next header: Inline
  .... ..00 .... = Hop limit: Inline (0x0)
  .... .... 0... .... = Context identifier extension: False
  .... .... .1.. .... = Source address compression: Stateful
  .... .... ..01 .... = Source address mode: 64-bits inline (0x0001)
  .... .... .... 0... = Multicast address compression: False
  .... .... .... .1.. = Destination address compression: Stateful
  .... .... .... ..01 = Destination address mode:
                                64-bits inline (0x0001)
  [Source context: bbbb::]
  [Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Hop limit: 128
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
Internet Protocol Version 6, Src: bbbb::1, Dst: bbbb::1415:92cc:0:3
0110 .... = Version: 6
.... 0000 0000 .... .... = Traffic Class:
                                0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... .... = Differentiated Services
                                Codepoint: Default (0)
.... .... ..00 .... .... = Explicit Congestion
                                Notification:
                                Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 40
Next Header: ICMPv6 (58)
Hop Limit: 128
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0xb65c [correct]
[Checksum Status: Good]
Identifier: 0x0001
Sequence: 63
[No response seen]
  [Expert Info (Warning/Sequence):
    No response seen to ICMPv6 request in frame 32291

```

Data (32 bytes)

Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
[Length: 32]

== Raw Bytes ==

```
0000  21 ec a3 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 f1 80 03 14 15 92 cc 00 00 00 02
0020  78 55 3a 80 00 00 00 00 00 00 01 14 15 92 cc
0030  00 00 00 03 80 00 b6 5c 00 01 00 3f 61 62 63 64
0040  65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74
0050  75 76 77 61 62 63 64 65 66 67 68 69 1e d3
```

[ping 3] ICMPv6 echo request 2->3

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:03,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
Long/64-bit

```
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..0. .... .... = Information Elements Present: False
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
```

Sequence Number: 94

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

FCS: 0x05ee (Correct)

6LoWPAN

IPHC Header

```
011. .... = Pattern: IP header compression (0x03)
...1 1... .... = Traffic class and flow label: Version,
                traffic class, and flow label compressed (0x3)
.... .0.. .... = Next header: Inline
.... ..00 .... = Hop limit: Inline (0x0)
.... .... 0... = Context identifier extension: False
.... .... .1.. .... = Source address compression: Stateful
```



```

..... ..01 ..... = Source address mode: 64-bits inline (0x0001)
..... ..0... = Multicast address compression: False
..... ..1.. = Destination address compression: Stateful
..... ..01 = Destination address mode:
                                                64-bits inline (0x0001)

[Source context: bbbb::]
[Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Hop limit: 128
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
Internet Protocol Version 6, Src: bbbb::1, Dst: bbbb::1415:92cc:0:3
0110 ..... = Version: 6
..... 0000 0000 ..... = Traffic Class: \
                                                0x00 (DSCP: CS0, ECN: Not-ECT)
..... 0000 00.. ..... = Differentiated Services
                                                Codepoint: Default (0)
..... ..00 ..... = Explicit Congestion
                                                Notification:
                                                Not ECN-Capable Transport (0)
..... ..0000 0000 0000 0000 0000 0000 = Flow Label: 0x000000
Payload Length: 40
Next Header: ICMPv6 (58)
Hop Limit: 128
Source: bbbb::1
Destination: bbbb::1415:92cc:0:3
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Echo (ping) request (128)
Code: 0
Checksum: 0xb65c [correct]
[Checksum Status: Good]
Identifier: 0x0001
Sequence: 63
[Response In: 3237]
Data (32 bytes)
  Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
  [Length: 32]

```

== Raw Bytes ==

```

0000  21 ec 5e fe ca 03 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 78 55 3a 80 00 00 00 00 00 00
0020  01 14 15 92 cc 00 00 00 03 80 00 b6 5c 00 01 00
0030  3f 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
0040  70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68

```

0050 69 ee 05

[ping 3] ICMPv6 echo reply 3->2

== Dissected packet ==

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                        Src: 14:15:92:cc:00:00:00:03
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0.... = Frame Pending: False
    .... .1.... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0.... = Sequence Number Suppression: False
    .... .0.... = Information Elements Present: False
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 177
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
FCS: 0x2455 (Correct)
6LoWPAN
    .... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
    100. .... = Routing Header 6lo: Critical Routing Header (0x04)
    ...0 .... = Packet direction (bit O): Up
    .... 0... = Rank-Error (bit R): No
    .... .0... = Forwarding-Error (bit F): No
    .... .1... = RPL Instance (bit I):
                                Elided (RPL Instance ID: 0)
    .... .0.... = Sender Rank Compression size (bit K): 2 bytes
    .... 0000 0101 = 6LoRH Type:
                                Routing Protocol Information (0x05)

    RPL Instance: 0x00
    Sender Rank: 0x039d
IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1... = Traffic class and flow label: Version,
                traffic class, and flow label compressed (0x3)
    .... .0... = Next header: Inline
    .... 10... = Hop limit: 64 (0x2)
    .... 0... = Context identifier extension: False

```

```

..... .1.. .... = Source address compression: Stateful
..... ..01 .... = Source address mode: 64-bits inline (0x0001)
..... .... 0... = Multicast address compression: False
..... .... .1.. = Destination address compression: Stateful
..... .... ..01 = Destination address mode:
                                     64-bits inline (0x0001)

[Source context: bbbb::]
[Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Source: bbbb::1415:92cc:0:3
Destination: bbbb::1
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:3, Dst: bbbb::1
0110 .... = Version: 6
.... 0000 0000 .... = Traffic Class:
                                     0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... = Differentiated Services
                                     Codepoint: Default (0)
.... .... ..00 .... = Explicit Congestion
                                     Notification:
                                     Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 40
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: bbbb::1415:92cc:0:3
Destination: bbbb::1
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Echo (ping) reply (129)
Code: 0
Checksum: 0xb55c [correct]
[Checksum Status: Good]
Identifier: 0x0001
Sequence: 63
[Response To: 3232]
[Response Time: 1913.163 ms]
Data (32 bytes)
  Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
  [Length: 32]

```

== Raw Bytes ==

```

0000  21 ec b1 fe ca 02 00 00 00 cc 92 15 14 03 00 00
0010  00 cc 92 15 14 f1 82 05 03 9d 7a 55 3a 14 15 92
0020  cc 00 00 00 03 00 00 00 00 00 00 00 01 81 00 b5
0030  5c 00 01 00 3f 61 62 63 64 65 66 67 68 69 6a 6b

```

```
0040  6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64
0050  65 66 67 68 69 55 24
```

[ping 3] ICMPv6 echo reply 2->1

== Dissected packet ==

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                    Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... 0... = Security Enabled: False
    .... .0... = Frame Pending: False
    .... .1... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0... = Sequence Number Suppression: False
    .... .0... = Information Elements Present: False
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    .... 10... = Frame Version: IEEE Std 802.15.4-2015 (2)
    .... 11... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 95
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
FCS: 0x9e34 (Correct)
6LoWPAN
    .... 0001 = Page Number: 1
6LoRH: Routing Protocol Information
    100. .... = Routing Header 6lo: Critical Routing Header (0x04)
    ...0 .... = Packet direction (bit O): Up
    .... 0... = Rank-Error (bit R): No
    .... .0... = Forwarding-Error (bit F): No
    .... .1... = RPL Instance (bit I):
                                                    Elided (RPL Instance ID: 0)
    .... .0... = Sender Rank Compression size (bit K): 2 bytes
    .... 0000 0101 = 6LoRH Type:
                                                    Routing Protocol Information (0x05)
RPL Instance: 0x00
Sender Rank: 0x026d
IPHC Header
    011. .... = Pattern: IP header compression (0x03)
    ...1 1... = Traffic class and flow label: Version,
                traffic class, and flow label compressed (0x3)
    .... .0... = Next header: Inline
    .... 10... = Hop limit: 64 (0x2)
```

```

..... 0... .. = Context identifier extension: False
..... .1.. .. = Source address compression: Stateful
..... ..01 .. = Source address mode: 64-bits inline (0x0001)
..... 0... .. = Multicast address compression: False
..... .1.. .. = Destination address compression: Stateful
..... ..01 .. = Destination address mode:
                                     64-bits inline (0x0001)
[Source context: bbbb::]
[Destination context: bbbb::]
Next header: ICMPv6 (0x3a)
Source: bbbb::1415:92cc:0:3
Destination: bbbb::1
Internet Protocol Version 6, Src: bbbb::1415:92cc:0:3, Dst: bbbb::1
0110 .... = Version: 6
.... 0000 0000 .... .. = Traffic Class:
                                     0x00 (DSCP: CS0, ECN: Not-ECT)
.... 0000 00.. .... .. = Differentiated Services
                                     Codepoint: Default (0)
.... .... ..00 .... .. = Explicit Congestion
                                     Notification:
                                     Not ECN-Capable Transport (0)
.... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
Payload Length: 40
Next Header: ICMPv6 (58)
Hop Limit: 64
Source: bbbb::1415:92cc:0:3
Destination: bbbb::1
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Echo (ping) reply (129)
Code: 0
Checksum: 0xb55c [correct]
[Checksum Status: Good]
Identifier: 0x0001
Sequence: 63
Data (32 bytes)
  Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
  [Length: 32]

```

== Raw Bytes ==

```

0000 21 ec 5f fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 f1 82 05 02 6d 7a 55 3a 14 15 92
0020 cc 00 00 00 03 00 00 00 00 00 00 00 01 81 00 b5
0030 5c 00 01 00 3f 61 62 63 64 65 66 67 68 69 6a 6b
0040 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64

```

0050 65 66 67 68 69 34 9e

4.8. 6P Commands and Response

4.8.1. 6P ADD

6P Command ADD 2->1

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
 Src: 14:15:92:cc:00:00:00:02
 Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
 Information Elements Present, Destination Addressing Mode: Long/64-bit,
 Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
 Long/64-bit

....001 = Frame Type: Data (0x1)
 0... = Security Enabled: False
0 = Frame Pending: False
1. = Acknowledge Request: True
0.. = PAN ID Compression: False
0 = Sequence Number Suppression: False
1. = Information Elements Present: True
 11.. = Destination Addressing Mode: Long/64-bit (0x3)
 ..10 = Frame Version: IEEE Std 802.15.4-2015 (2)
 11.. = Source Addressing Mode: Long/64-bit (0x3)

Sequence Number: 0

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Header IEs, Header Termination 1 IE

Header Termination 1 IE (Payload IEs follow)

IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
 Length: 0

0... = Type: Header (0)
 .011 1111 0... = Id: Header Termination 1 IE (0x7e)
000 0000 = Length: 0

Payload IEs, IETF IE

IETF Payload IE

Payload IE TLV: 0xa81d, Type: Payload, Id: IETF IE

1... = Type: Payload (1)
 .010 1... = Id: IETF IE (0x5)
000 0001 1101 = Length: 29

Sub-ID: 201

6top IE

.... 0000 = 6P Version: 0
 ..00 = Type: Request (0x0)
 00.. = Reserved: 0x0
 Code: 0x01 (ADD)
 SFID (6top Scheduling Function ID): 0x00

```

..... 0000 = SeqNum: 0
0000 ..... = GEN: Clear (0)
Metadata: 0x0000
Cell Options: TX|RX|SHARED (0x07)
..... ..1 = Transmit (TX) Cell: 0x1
..... ..1. = Receive (RX) Cell: 0x1
..... .1.. = SHARED Cell: 0x1
0000 0... = Reserved: 0x00
Number of Cells: 1
CellList
  Cell: 3d000600
    Slot Offset: 0x003d
    Channel Offset: 0x0006
  Cell: 08000400
    Slot Offset: 0x0008
    Channel Offset: 0x0004
  Cell: 17000f00
    Slot Offset: 0x0017
    Channel Offset: 0x000f
  Cell: 3e000600
    Slot Offset: 0x003e
    Channel Offset: 0x0006
  Cell: 29000900
    Slot Offset: 0x0029
    Channel Offset: 0x0009
FCS: 0xd5e5 (Correct)
== Raw Bytes ==

0000 21 ee 00 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 00 3f 1d a8 c9 00 01 00 00 00 00
0020 07 01 3d 00 06 00 08 00 04 00 17 00 0f 00 3e 00
0030 06 00 29 00 09 00 e5 d5

```

6P Response to ADD 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                      Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                    Long/64-bit
..... ..001 = Frame Type: Data (0x1)
..... ..0... = Security Enabled: False
..... ...0 .... = Frame Pending: False
..... ...1. .... = Acknowledge Request: True
..... ...0.. .... = PAN ID Compression: False
..... ...0 .... = Sequence Number Suppression: False

```

```

.....1. .... = Information Elements Present: True
.... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 97
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Header IEs, Header Termination 1 IE
  Header Termination 1 IE (Payload IEs follow)
    IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                    Length: 0
      0... .... = Type: Header (0)
      .011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
      .... .... .000 0000 = Length: 0
Payload IEs, IETF IE
  IETF Payload IE
    Payload IE TLV: 0xa809, Type: Payload, Id: IETF IE
      1... .... = Type: Payload (1)
      .010 1... .... = Id: IETF IE (0x5)
      .... .000 0000 1001 = Length: 9
    Sub-ID: 201
    6top IE
      .... 0000 = 6P Version: 0
      ..01 .... = Type: Response (0x1)
      00.. .... = Reserved: 0x0
      Code: 0x00 (SUCCESS)
      SFID (6top Scheduling Function ID): 0x00
      .... 0000 = SeqNum: 0
      0000 .... = GEN: Clear (0)
      CellList
        Cell: 3d000600
          Slot Offset: 0x003d
          Channel Offset: 0x0006
FCS: 0xc934 (Correct)

```

== Raw Bytes ==

```

0000 21 ee 61 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 09 a8 c9 10 00 00 00 3d 00
0020 06 00 34 c9

```

4.8.2. 6P COUNT

6P Command COUNT 2->1

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,

Src: 14:15:92:cc:00:00:00:02

Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

Long/64-bit

```

.....001 = Frame Type: Data (0x1)
.....0... = Security Enabled: False
.....0.... = Frame Pending: False
.....1.... = Acknowledge Request: True
.....0... = PAN ID Compression: False
.....0.... = Sequence Number Suppression: False
.....1.... = Information Elements Present: True
....11.... = Destination Addressing Mode: Long/64-bit (0x3)
..10.... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.... = Source Addressing Mode: Long/64-bit (0x3)

```

Sequence Number: 22

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Header IEs, Header Termination 1 IE

Header Termination 1 IE (Payload IEs follow)

IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
Length: 0

```

0... = Type: Header (0)
.011 1111 0... = Id: Header Termination 1 IE (0x7e)
.... .000 0000 = Length: 0

```

Payload IEs, IETF IE

IETF Payload IE

Payload IE TLV: 0xa808, Type: Payload, Id: IETF IE

```

1... = Type: Payload (1)
.010 1... = Id: IETF IE (0x5)
.... .000 0000 1000 = Length: 8

```

Sub-ID: 201

6top IE

```

.... 0000 = 6P Version: 0
..00.... = Type: Request (0x0)
00... = Reserved: 0x0
Code: 0x04 (COUNT)
SFID (6top Scheduling Function ID): 0x00
.... 0010 = SeqNum: 2
0000.... = GEN: Clear (0)
Metadata: 0x0000
Cell Options: TX (0x01)
.... .1 = Transmit (TX) Cell: 0x1
.... .0. = Receive (RX) Cell: 0x0
.... .0.. = SHARED Cell: 0x0
0000 0... = Reserved: 0x00

```

FCS: 0x1fb7 (Correct)

== Raw Bytes ==

```
0000  21 ee 16 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 00 3f 08 a8 c9 00 04 00 02 00 00
0020  01 b7 1f
```

6P Response to COUNT 1->2

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
 Src: 14:15:92:cc:00:00:00:01
 Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
 Information Elements Present, Destination Addressing Mode: Long/64-bit,
 Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
 Long/64-bit

```
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..1. .... .... = Information Elements Present: True
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
...10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
```

Sequence Number: 104

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Header IEs, Header Termination 1 IE

Header Termination 1 IE (Payload IEs follow)

```
IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
Length: 0
```

```
0... .... .... .... = Type: Header (0)
.011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
.... .... .000 0000 = Length: 0
```

Payload IEs, IETF IE

IETF Payload IE

Payload IE TLV: 0xa807, Type: Payload, Id: IETF IE

```
1... .... .... .... = Type: Payload (1)
.010 1... .... .... = Id: IETF IE (0x5)
.... .000 0000 0111 = Length: 7
```

Sub-ID: 201

6top IE

```
.... 0000 = 6P Version: 0
..01 .... = Type: Response (0x1)
00.. .... = Reserved: 0x0
Code: 0x00 (SUCCESS)
SFID (6top Scheduling Function ID): 0x00
```

```

        .... 0010 = SeqNum: 2
        0000 .... = GEN: Clear (0)
        Total Number of Cells: 0
FCS: 0x6ca9 (Correct)
== Raw Bytes ==

```

```

0000  21 ee 68 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 00 3f 07 a8 c9 10 00 00 02 00 00
0020  a9 6c

```

4.8.3. 6P DELETE

6P Command DELETE 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
                      Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                Long/64-bit
    .... .... .... .001 = Frame Type: Data (0x1)
    .... .... .... 0... = Security Enabled: False
    .... .... ...0 .... = Frame Pending: False
    .... .... ..1. .... = Acknowledge Request: True
    .... .... .0.. .... = PAN ID Compression: False
    .... ...0 .... .... = Sequence Number Suppression: False
    .... ..1. .... .... = Information Elements Present: True
    .... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 46
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Header IEs, Header Termination 1 IE (Payload IEs follow)
    Header Termination 1 IE (Payload IEs follow)
        IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                Length: 0
        0... .... .... .... = Type: Header (0)
        .011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
        .... .... .000 0000 = Length: 0
Payload IEs, IETF Payload IE
    IETF Payload IE
        IE Header: 0xa80d, Type: Payload, Id: IETF IE, Length: 13
        1... .... .... .... = Type: Payload (1)
        .010 1... .... .... = Id: IETF IE (0x5)
        .... .000 0000 1101 = Length: 13
Sub-ID: 201

```

```

6top IE
.... 0000 = 6P Version: 0
..00 .... = Type: Request (0x0)
00.. .... = Reserved: 0x0
Code: 0x02 (DELETE)
SFID (6top Scheduling Function ID): 0x00
1011 1110 = SeqNum: 190
Metadata: 0x0000
Cell Options: TX (0x01)
.... ...1 = Transmit (TX) Cell: 0x1
.... ..0. = Receive (RX) Cell: 0x0
.... .0.. = SHARED Cell: 0x0
0000 0... = Reserved: 0x00
Number of Cells: 1
CellList
  Cell: 13000700
    Slot Offset: 0x0013
    Channel Offset: 0x0007
FCS: 0x5843 (Correct)

== Raw Bytes ==

0000  21 ee 2e fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 00 3f 0d a8 c9 00 02 00 be 00 00
0020  01 01 13 00 07 00 43 58

6P Response to DELETE 1->2

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                      Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                    Long/64-bit
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..1. .... .... = Information Elements Present: True
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 107
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

```

```

Header IEs, Header Termination 1 IE (Payload IEs follow)
  Header Termination 1 IE (Payload IEs follow)
    IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                Length: 0
      0... .. = Type: Header (0)
      .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
      .... .. .000 0000 = Length: 0
Payload IEs, IETF Payload IE
  IETF Payload IE
    IE Header: 0xa809, Type: Payload, Id: IETF IE, Length: 9
      1... .. = Type: Payload (1)
      .010 1... .. = Id: IETF IE (0x5)
      .... .000 0000 1001 = Length: 9
    Sub-ID: 201
    6top IE
      .... 0000 = 6P Version: 0
      ..01 .... = Type: Response (0x1)
      00.. .... = Reserved: 0x0
      Code: 0x00 (SUCCESS)
      SFID (6top Scheduling Function ID): 0x00
      1011 1110 = SeqNum: 190
      CellList
        Cell: 13000700
          Slot Offset: 0x0013
          Channel Offset: 0x0007
FCS: 0x8326 (Correct)

```

== Raw Bytes ==

```

0000  21 ee 6b fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010  00 cc 92 15 14 00 3f 09 a8 c9 10 00 00 be 13 00
0020  07 00 26 83

```

4.8.4. 6P RELOCATE

6P Command RELOCATE 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                Long/64-bit
  .... .. .001 = Frame Type: Data (0x1)
  .... .. 0... = Security Enabled: False
  .... .. .0 .... = Frame Pending: False
  .... .. .1. .... = Acknowledge Request: True
  .... .. .0.. .... = PAN ID Compression: False

```

```

.....0..... = Sequence Number Suppression: False
.....1..... = Information Elements Present: True
.....11..... = Destination Addressing Mode: Long/64-bit (0x3)
..10..... = Frame Version: IEEE Std 802.15.4-2015 (2)
11..... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 121
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Header IEs, Header Termination 1 IE (Payload IEs follow)
  Header Termination 1 IE (Payload IEs follow)
    IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                    Length: 0
      0.... = Type: Header (0)
      .011 1111 0... = Id: Header Termination 1 IE (0x7e)
      .... .000 0000 = Length: 0
Payload IEs, IETF Payload IE
IETF Payload IE
  IE Header: 0xa819, Type: Payload, Id: IETF IE, Length: 25
    1.... = Type: Payload (1)
    .010 1... = Id: IETF IE (0x5)
    .... .000 0001 1001 = Length: 25
Sub-ID: 201
6top IE
  .... 0000 = 6P Version: 0
  ..00 .... = Type: Request (0x0)
  00.. .... = Reserved: 0x0
  Code: 0x03 (RELOCATE)
  SFID (6top Scheduling Function ID): 0x00
  0011 0010 = SeqNum: 50
  Metadata: 0x0000
  Cell Options: TX (0x01)
    .... .1 = Transmit (TX) Cell: 0x1
    .... .0. = Receive (RX) Cell: 0x0
    .... .0.. = SHARED Cell: 0x0
    0000 0... = Reserved: 0x00
  Number of Cells: 1
  Rel. CellList
    Cell: 11000900
      Slot Offset: 0x0011
      Channel Offset: 0x0009
  Cand. CellList
    Cell: 19000700
      Slot Offset: 0x0019
      Channel Offset: 0x0007
    Cell: 16000500
      Slot Offset: 0x0016
      Channel Offset: 0x0005

```

Cell: 14000300
 Slot Offset: 0x0014
 Channel Offset: 0x0003
 FCS: 0xadd3 (Correct)

== Raw Bytes ==

```
0000  21 ee 79 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 00 3f 19 a8 c9 00 03 00 32 00 00
0020  01 01 11 00 09 00 19 00 07 00 16 00 05 00 14 00
0030  03 00 d3 ad
```

6P Response to RELOCATE 1->2

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
 Src: 14:15:92:cc:00:00:00:01
 Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
 Information Elements Present, Destination Addressing Mode: Long/64-bit,
 Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
 Long/64-bit

```
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... .... 0... = Frame Pending: False
.... .... .... 1... = Acknowledge Request: True
.... .... .0... = PAN ID Compression: False
.... .... 0... = Sequence Number Suppression: False
.... .... 1... = Information Elements Present: True
.... 11... = Destination Addressing Mode: Long/64-bit (0x3)
...10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11... .... = Source Addressing Mode: Long/64-bit (0x3)
```

Sequence Number: 205

Destination PAN: 0xcafe

Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Header IEs, Header Termination 1 IE (Payload IEs follow)

Header Termination 1 IE (Payload IEs follow)

IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
 Length: 0

```
0... .... = Type: Header (0)
.011 1111 0... = Id: Header Termination 1 IE (0x7e)
.... .... .000 0000 = Length: 0
```

Payload IEs, IETF Payload IE

IETF Payload IE

IE Header: 0xa809, Type: Payload, Id: IETF IE, Length: 9

```
1... .... = Type: Payload (1)
.010 1... = Id: IETF IE (0x5)
.... .000 0000 1001 = Length: 9
```

```

Sub-ID: 201
6top IE
.... 0000 = 6P Version: 0
..01 .... = Type: Response (0x1)
00.. .... = Reserved: 0x0
Code: 0x00 (SUCCESS)
SFID (6top Scheduling Function ID): 0x00
0011 0010 = SeqNum: 50
CellList
    Cell: 19000700
        Slot Offset: 0x0019
        Channel Offset: 0x0007
FCS: 0x6784 (Correct)

```

== Raw Bytes ==

```

0000 21 ee cd fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 09 a8 c9 10 00 00 32 19 00
0020 07 00 84 67

```

4.8.5. 6P LIST

6P Command LIST 2->1

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                                    Long/64-bit
.... .... .... .001 = Frame Type: Data (0x1)
.... .... .... 0... = Security Enabled: False
.... .... ...0 .... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... ...0 .... .... = Sequence Number Suppression: False
.... ..1. .... .... = Information Elements Present: True
.... 11.. .... .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 99
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Header IEs, Header Termination 1 IE (Payload IEs follow)
    Header Termination 1 IE (Payload IEs follow)
        IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                                    Length: 0

```



```

0... .. = Type: Header (0)
.011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
.... .. .000 0000 = Length: 0
Payload IEs, IETF Payload IE
  IETF Payload IE
    IE Header: 0xa80d, Type: Payload, Id: IETF IE, Length: 13
    1... .. = Type: Payload (1)
    .010 1... .. = Id: IETF IE (0x5)
    .... .000 0000 1101 = Length: 13
  Sub-ID: 201
  6top IE
    .... 0000 = 6P Version: 0
    ..00 .... = Type: Request (0x0)
    00.. .... = Reserved: 0x0
    Code: 0x05 (LIST)
    SFID (6top Scheduling Function ID): 0x00
    1000 1011 = SeqNum: 139
    Metadata: 0x0000
    Cell Options: TX (0x01)
      .... ..1 = Transmit (TX) Cell: 0x1
      .... ..0 = Receive (RX) Cell: 0x0
      .... .0.. = SHARED Cell: 0x0
      0000 0... = Reserved: 0x00
    Reserved: 0x00
    Offset: 1
    Maximum Number of Requested Cells: 4
FCS: 0x5fdd (Correct)

```

== Raw Bytes ==

```

0000 21 ee 63 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010 00 cc 92 15 14 00 3f 0d a8 c9 00 05 00 8b 00 00
0020 01 00 01 00 04 00 dd 5f

```

6P Response to LIST 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
Long/64-bit

```

```

.... .. .001 = Frame Type: Data (0x1)
.... .. 0... = Security Enabled: False
.... .. ..0 .... = Frame Pending: False
.... .. ..1. .... = Acknowledge Request: True
.... .. ..0.. .... = PAN ID Compression: False
.... .. ..0 .... = Sequence Number Suppression: False

```

```

.....1. .... = Information Elements Present: True
.....11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 207
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Header IEs, Header Termination 1 IE (Payload IEs follow)
  Header Termination 1 IE (Payload IEs follow)
    IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                    Length: 0
      0... .... = Type: Header (0)
      .011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
      .... .... .000 0000 = Length: 0
Payload IEs, IETF Payload IE
  IETF Payload IE
    IE Header: 0xa811, Type: Payload, Id: IETF IE, Length: 17
      1... .... = Type: Payload (1)
      .010 1... .... = Id: IETF IE (0x5)
      .... .000 0001 0001 = Length: 17
    Sub-ID: 201
    6top IE
      .... 0000 = 6P Version: 0
      ..01 .... = Type: Response (0x1)
      00.. .... = Reserved: 0x0
      Code: 0x01 (RC_EOL)
      SFID (6top Scheduling Function ID): 0x00
      1000 1011 = SeqNum: 139
      CellList
        Cell: 41000800
          Slot Offset: 0x0041
          Channel Offset: 0x0008
        Cell: 3c000700
          Slot Offset: 0x003c
          Channel Offset: 0x0007
        Cell: 19000700
          Slot Offset: 0x0019
          Channel Offset: 0x0007
FCS: 0x7594 (Correct)

== Raw Bytes ==

0000  21 ee 65 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010  00 cc 92 15 14 00 3f 11 a8 c9 00 02 00 8c 00 00
0020  07 01 3c 00 07 00 19 00 07 00 05 64

```

4.8.6. 6P CLEAR

6P Command CLEAR 2->1

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02

Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:

Long/64-bit

```

.... .... .001 = Frame Type: Data (0x1)
.... .... 0... = Security Enabled: False
.... .... ..0... = Frame Pending: False
.... .... ..1. .... = Acknowledge Request: True
.... .... .0.. .... = PAN ID Compression: False
.... .... 0 .... = Sequence Number Suppression: False
.... .... ..1. .... = Information Elements Present: True
.... 11.. .... = Destination Addressing Mode: Long/64-bit (0x3)
..10 .... .... = Frame Version: IEEE Std 802.15.4-2015 (2)
11.. .... .... = Source Addressing Mode: Long/64-bit (0x3)

```

Sequence Number: 181

Destination PAN: 0xc4fe

Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)

Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)

Header IEs, Header Termination 1 IE (Payload IEs follow)

Header Termination 1 IE (Payload IEs follow)

IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
Length: 0

```

0... .... = Type: Header (0)
.011 1111 0... .... = Id: Header Termination 1 IE (0x7e)
.... .... .000 0000 = Length: 0

```

Payload IEs, IETF Payload IE

IETF Payload IE

IE Header: 0xa807, Type: Payload, Id: IETF IE, Length: 7

```

1... .... = Type: Payload (1)
.010 1... .... = Id: IETF IE (0x5)
.... .000 0000 0111 = Length: 7

```

Sub-ID: 201

6top IE

```

.... 0000 = 6P Version: 0
..00 .... = Type: Request (0x0)
00.. .... = Reserved: 0x0
Code: 0x07 (CLEAR)
SFID (6top Scheduling Function ID): 0x00
0101 0001 = SeqNum: 81
Metadata: 0x0000

```

FCS: 0x0e2c (Correct)

== Raw Bytes ==

```
0000    21 ee b5 fe ca 01 00 00 00 cc 92 15 14 02 00 00
0010    00 cc 92 15 14 00 3f 07 a8 c9 00 07 00 51 00 00
0020    2c 0e
```

6P Response to CLEAR 1->2

```

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
                      Src: 14:15:92:cc:00:00:00:01
Frame Control Field: 0xee21, Frame Type: Data, Acknowledge Request,
Information Elements Present, Destination Addressing Mode: Long/64-bit,
Frame Version: IEEE Std 802.15.4-2015, Source Addressing Mode:
                                                    Long/64-bit
    .... .001 = Frame Type: Data (0x1)
    .... .0... = Security Enabled: False
    .... .0.... = Frame Pending: False
    .... .1.... = Acknowledge Request: True
    .... .0... = PAN ID Compression: False
    .... .0.... = Sequence Number Suppression: False
    .... .1.... = Information Elements Present: True
    .... 11... = Destination Addressing Mode: Long/64-bit (0x3)
    ..10 .... = Frame Version: IEEE Std 802.15.4-2015 (2)
    11... .. = Source Addressing Mode: Long/64-bit (0x3)
Sequence Number: 185
Destination PAN: 0xcafe
Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
Header IEs, Header Termination 1 IE (Payload IEs follow)
    Header Termination 1 IE (Payload IEs follow)
        IE Header: 0x3f00, Type: Header, Id: Header Termination 1 IE,
                                                    Length: 0
            0... .. = Type: Header (0)
            .011 1111 0... .. = Id: Header Termination 1 IE (0x7e)
            .... .000 0000 = Length: 0
Payload IEs, IETF Payload IE
    IETF Payload IE
        IE Header: 0xa805, Type: Payload, Id: IETF IE, Length: 5
            1... .. = Type: Payload (1)
            .010 1... .. = Id: IETF IE (0x5)
            .... .000 0000 0101 = Length: 5
        Sub-ID: 201
        6top IE
            .... 0000 = 6P Version: 0
            ..01 .... = Type: Response (0x1)
            00.. .... = Reserved: 0x0
            Code: 0x00 (SUCCESS)
            SFID (6top Scheduling Function ID): 0x00
            0101 0001 = SeqNum: 81
FCS: 0x3fe0 (Correct)

== Raw Bytes ==
0000 21 ee b9 fe ca 02 00 00 00 cc 92 15 14 01 00 00
0010 00 cc 92 15 14 00 3f 05 a8 c9 10 00 00 51 e0 3f

```

5. [TEMPORARY] Known Bugs/Issues

This document tracks the standardization activity, and reflects the state of the implementation. This document is updated regularly. Sometimes, the [OpenWSN] implementation falls behind on the standardization. In this section, we list the known issues or the elements that are not implemented. This section will be removed when the final version of the document is produced.

- o All link-layer frames are presented without link-layer security. This will be fixed in future revisions, both types of frames will then be shown: secured (what actually goes over the air) and unsecured (which Wireshark can parse).
- o ICMPv6 echo request packets use source and destination global addresses but their reply packets use link local addresses.

6. IANA Considerations

This memo includes no requests to IANA.

7. Security Considerations

This memo only presents example frames exchanged. It does not define any protocol; there are hence no security considerations in this document.

8. Acknowledgments

The authors would like to thank the OpenWSN community, the 6TiSCH working group and the participants at the 6TiSCH plugtests for their feedback which has helped shape this document.

9. References

9.1. Normative References

[I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., Pister, K., and M. Richardson,
"Minimal Security Framework for 6TiSCH", draft-ietf-
6tisch-minimal-security-06 (work in progress), May 2018.

9.2. External Informative References

[OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.

Authors' Addresses

Jonathan Munoz (editor)
Inria
2 rue Simone Iff
Paris 12 75012
France

Email: jonathan.munoz@inria.fr

Dominique Barthel
Orange Labs
28 Chemin du Vieux Chene
Meylan 38240
France

Email: dominique.barthel@orange.com

6TiSCH
Internet-Draft
Intended status: Informational
Expires: January 3, 2018

G. Papadopoulos, Ed.
N. Montavont
IMT Atlantique
P. Thubert
Cisco
July 2, 2017

Exploiting Packet Replication and Elimination in Complex Tracks in
6TiSCH LLNs
draft-papadopoulos-6tisch-pre-reqs-00

Abstract

6TiSCH Packet Replication and Elimination mechanism consists in duplicating data packets into several paths in the network to increase reliability and provide low jitter. Over a wireless medium, this technique can take advantage of communication overhearing, when parallel transmissions over two adjacent paths are scheduled. This document presents the concept and details the required changes to the current specifications that will be necessary to enable this.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 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	2
2. Terminology	3
3. Tracks	3
3.1. Tracks Overview	3
3.2. Complex Tracks	3
4. Packet Replication and Elimination principles	3
4.1. Packet Replication	4
4.2. Packet Elimination	5
4.3. Promiscuous Overhearing	5
5. Requirements	6
5.1. Requirements Related to Alternative Parent Selection . .	6
5.2. Requirements Related to Promiscuous Overhearing	6
5.3. Requirements Related to Cells without ACKs	7
5.4. Requirements Related to Packet Elimination	7
6. Security Considerations	7
7. IANA Considerations	7
8. References	7
8.1. Informative references	8
8.2. Other Informative References	8
Authors' Addresses	8

1. Introduction

Some applications (such as Wireless Industrial IoT) require robust communications framework that guarantees data packet delivery in a given delay. For example, a periodic process may need to be repeated identically every time. To reach this ambition, the network must not only be reliable but also deterministic.

A deterministic network ensures that the transported data packet will be carried out in a pre-defined and in a tight window of time, whatever the quality of the wireless links and the network congestion. The goal of such network is to exhibit ultra-low jitter performance, i.e., close to 0. IEEE std. 802.15.4 [IEEE802154-2015] has provision to provide guarantees for deterministic networks. Time-Slotted Channel Hopping (TSCH) provides transmission schedule to avoid random access to the medium and channel diversity to fight interferences. However, TSCH is prone to retransmissions when the actual transmission was unsuccessful, due to external interference or

potential collision and, consequently, it increases the end-to-end delay performance.

This document is mainly motivated by the ongoing work in the 6TiSCH working group. The architecture of a 6TiSCH network is detailed in 6TiSCH Architecture [I-D.ietf-6tisch-architecture] draft, which is used for the remainder of this document. In this specification, we focus on Complex Track with Replication and Elimination.

2. Terminology

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

3. Tracks

3.1. Tracks Overview

The 6TiSCH architecture introduces the concept of Tracks in 6TiSCH Architecture [I-D.ietf-6tisch-architecture]. A simple track is composed of a sequence of cells (a combination of a transmitter, a receiver and a given channel offset) to ensure the transmission of a single packet from a source 6TiSCH node to a destination 6TiSCH node across a 6TiSCH multihop path.

3.2. Complex Tracks

A Complex Track is designed as a directed acyclic graph from a source 6TiSCH node towards a destination 6TiSCH node to support multi-path forwarding, as introduced in 6TiSCH Architecture [I-D.ietf-6tisch-architecture]. By employing DetNet Packet Replication and Elimination (PRE) techniques, several paths may be computed, and these paths may be more or less independent. For example, a complex Track may branch off and rejoin over non-congruent paths (branches).

In the following Section, we will detail Deterministic Networks PRE techniques.

4. Packet Replication and Elimination principles

In a nutshell, PRE consists in establishing several paths in a network to provide redundancy and parallel transmissions to bound the delay to traverse the network. Optionnally, promiscuous listening between paths is possible, such that the nodes on one path may overhear transmissions along the other path. Considering the scenario depicted in Figure 1, many different paths are possible for

S to reach R. A simple way to take benefit from this topology could be to use the two independent paths via nodes A, C, E and via B, D, F. But more complex paths are possible by interleaving transmissions from one level of the path to the upper level in a ship-in-the-night fashion. The 6TiSCH PRE may also take advantage to the shared properties of the medium to compensate for the potential loss that is incurred with radio transmissions. For instance, when the source sends to A, B may listen and get a second chance to receive the frame without an additional transmission. Note that B would not have to listen if it already received that particular frame at an earlier time slot.

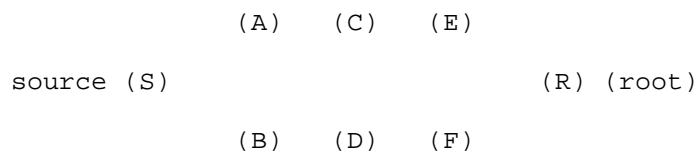


Figure 1: A Typical Ladder Shape with Two Parallel Paths Toward the Destination

PRE model can be implemented in both centralized and distributed scheduling approach. In the centralized approach, a scheduler calculates the routes and schedules the communication among the nodes along a circuit such as a Label switched path. In the distributed approach, each node selects its route to the destination. In both cases, a default parent and alternate parent(s) should be selected to set up complex tracks.

In the following Subsections, detailed description of all required operations defined by PRE, namely, Alternative Path Selection, Packet Replication, Packet Elimination and Promiscuous Overhearing, will be described.

4.1. Packet Replication

The objective of PRE is to offer deterministic networking properties, with high reliability and bounded latency. To achieve this goal, determinism in every level of the forwarding path should be guaranteed. By employing Packet Replication procedure, each node transmits (i.e., replicates) each data packet to both its Default Parent (DP) and Alternative Parent (AP). To do so, each node (i.e., source and intermediate 6TiSCH nodes) transmits the data packet twice in unicast to each parent. For instance, in Figure 2, the source 6TiSCH node S is transmitting the packet to both parents, nodes A and

B, in two different timeslots within the same TSCH slotframe. Thus, the packet eventually obtains parallel paths to the destination.

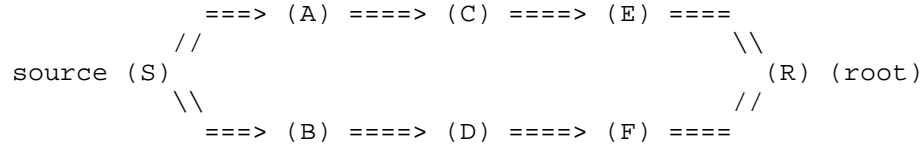


Figure 2: Packet Replication: S transmits twice the same data packet, to its DP (A) and to its AP (B).

4.2. Packet Elimination

The replication operation increases the traffic load in the network, due to packet duplications. Thus, Packet Elimination operation should be applied at each RPL DAG level to reduce the unnecessary traffic. To this aim, once a node receives the first copy of a data packet, it discards the following copies. Because the first copy that reaches a node is the one that counts, and thus will be the only copy that will be forwarded upward.

4.3. Promiscuous Overhearing

Considering that the wireless medium is broadcast by nature, any neighbor of a transmitter may overhear a transmission. By employing the Promiscuous Overhearing operation, DP and AP eventually have more chances to receive the data packets. In Figure 3, when node A is transmitting to its DP (node C), the AP (node D) and its Sibling (node B) may decode this data packet as well. As a result, by employing correlated paths, a node may have multiple opportunities to receive a given data packet. This feature not only enhances the end-to-end reliability but also it reduces the end-to-end delay.

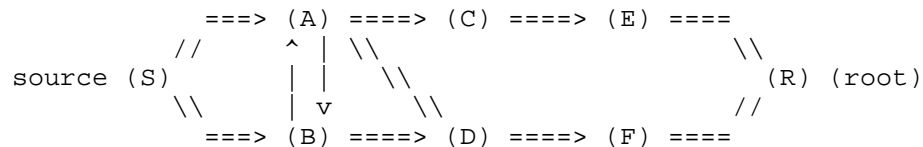


Figure 3: Unicast to DP with Overhearing: by employing Promiscuous Overhearing, DP, AP and the Sibling nodes have more opportunities to receive the same data packet.

5. Requirements

5.1. Requirements Related to Alternative Parent Selection

To perform the Replication procedure, it is necessary to define the Alternative Parent(s) and, consequently, the path to the destination 6TiSCH node, for each node in the 6TiSCH network. An AP can be selected in many different ways, and is dependent on the implementation. However, control packets should give some metrics to discriminate between different neighbors.

Related requirements are:

Req1.1: To design such algorithm, RPL DODAG Information Object (DIO) message format SHOULD be extended with an option to allow for a 6TiSCH node to learn additional information for its potential parent and its list of parents.

Req1.2: The routing protocol SHOULD be extended to allow for each 6TiSCH node to select AP(s) and duplicate a packet to several next hops.

5.2. Requirements Related to Promiscuous Overhearing

As stated previously, to further increase the 6TiSCH network reliability and to achieve deterministic packet deliveries at the destination 6TiSCH node, promiscuous overhearing can be considered.

As it is described in BCP 210 [RFC8180], in TSCH mode, the data frames are transmitted in unicast mode and are acknowledged by the receiving neighbor. To perform the promiscuous overhearing procedure, there SHOULD be an option for the transmitted frames, i.e., in unicast, to be overheard by the potential neighborhood 6TiSCH node.

Related requirements are:

Req2.1: The 6top Protocol [I-D.ietf-6tisch-6top-protocol] SHOULD be extended to allow optionally a cell reservation with two receivers, i.e., DP and AP. Considering that each frame may be transmitted twice in unicast to each parent, then depending the transmission, either DP will acknowledge the frame or AP will.

Req2.2: Next, to request the overhearing cells, the 6P ADD Request Format SHOULD be transmitted either twice to each parent, i.e., DP and AP, or once in multicast to both parents.

5.3. Requirements Related to Cells without ACKs

As stated in BCP 210 [RFC8180], each data frame is acknowledged by the receiving 6TiSCH node. However, by employing promiscuous overhearing operation, particular attention should be given to who will acknowledge a transmission, i.e., the DP, and / or one of the AP(s)

Related requirements are:

Req4.1: To avoid the ACK collision, the TSCH Schedule as per BCP 210 [RFC8180], only the DP MUST acknowledge the data packet.

Req4.2: Alternatively, to achieve further consistency the overheard transmission need be acknowledged by both parents, i.e., DP and AP. To do so, BCP 210 [RFC8180] SHOULD be extended accordingly.

5.4. Requirements Related to Packet Elimination

By employing packet replication operation, the 6TiSCH network expects to perform the packet elimination operation along a complex Track to bound the number of the duplicated packets, i.e., the unnecessary traffic.

Related requirements are:

Req5.1: As per 6TiSCH Architecture [I-D.ietf-6tisch-architecture], 6TiSCH has no position about how the sequence numbers would be tagged in the packet. However, it comes with Tagging Packets for Flow Identification. More specifically, a 6TiSCH network expects that timeslots corresponding to copies of a same frame along a complex Track are correlated by configuration and, thus, does not need to process the sequence numbers.

6. Security Considerations

TODO.

7. IANA Considerations

This document has no IANA considerations.

8. References

8.1. Informative references

- [I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-07 (work in progress), June 2017.
- [I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-11 (work in progress), January 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<http://www.rfc-editor.org/info/rfc6550>>.
- [RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180, May 2017, <<http://www.rfc-editor.org/info/rfc8180>>.

8.2. Other Informative References

- [IEEE802154-2015]
IEEE standard for Information Technology, "IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015".

Authors' Addresses

Georgios Papadopoulos (editor)
IMT Atlantique
Office B00 - 102A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 04
Email: georgios.papadopoulos@imt-atlantique.fr

Nicolas Montavont
IMT Atlantique
Office B00 - 106A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 23
Email: nicolas.montavont@imt-atlantique.fr

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

6TiSCH
Internet-Draft
Intended status: Informational
Expires: January 27, 2019

G. Papadopoulos, Ed.
N. Montavont
IMT Atlantique
P. Thubert
Cisco
July 26, 2018

Exploiting Packet Replication and Elimination in Complex Tracks in
6TiSCH LLNs
draft-papadopoulos-6tisch-pre-reqs-02

Abstract

6TiSCH Packet Replication and Elimination mechanism consists in duplicating data packets into several paths in the network to increase reliability and provide low jitter. Over a wireless medium, this technique can take advantage of communication overhearing, when parallel transmissions over two adjacent paths are scheduled. This document presents the concept and details the required changes to the current specifications that will be necessary to enable this.

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 January 27, 2019.

Copyright Notice

Copyright (c) 2018 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. Terminology	3
3. Tracks	3
3.1. Tracks Overview	3
3.2. Complex Tracks	3
4. Packet Replication and Elimination principles	3
4.1. Packet Replication	4
4.2. Packet Elimination	5
4.3. Promiscuous Overhearing	5
5. Requirements	6
5.1. Requirements Related to Alternative Parent Selection . .	6
5.2. Requirements Related to Propagated Information	6
5.3. Requirements Related to Cell Reservation	7
5.4. Requirements Related to Cells without ACKs	8
5.5. Requirements Related to Packet Elimination	9
6. Security Considerations	9
7. IANA Considerations	9
8. References	9
8.1. Informative references	9
8.2. Other Informative References	10
Authors' Addresses	10

1. Introduction

Some applications (such as Wireless Industrial IoT) require robust communications framework that guarantees data packet delivery in a given delay. For example, a periodic process may need to be repeated identically every time. To reach this ambition, the network must not only be reliable but also deterministic.

A deterministic network ensures that the transported data packet will be carried out in a pre-defined and in a tight window of time, whatever the quality of the wireless links and the network congestion. The goal of such network is to exhibit ultra-low jitter performance, i.e., close to 0. IEEE std. 802.15.4 [IEEE802154-2015] has provision to provide guarantees for deterministic networks. Time-Slotted Channel Hopping (TSCH) provides transmission schedule to avoid random access to the medium and channel diversity to fight interferences. However, TSCH is prone to retransmissions when the actual transmission was unsuccessful, due to external interference or

potential collision and, consequently, it increases the end-to-end delay performance.

This document is mainly motivated by the ongoing work in the 6TiSCH working group. The architecture of a 6TiSCH network is detailed in 6TiSCH Architecture [I-D.ietf-6tisch-architecture] draft, which is used for the remainder of this document. In this specification, we focus on Complex Track with Replication and Elimination.

2. Terminology

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

3. Tracks

3.1. Tracks Overview

The 6TiSCH architecture introduces the concept of Tracks in 6TiSCH Architecture [I-D.ietf-6tisch-architecture]. A simple track is composed of a sequence of cells (a combination of a transmitter, a receiver and a given channel offset) to ensure the transmission of a single packet from a source 6TiSCH node to a destination 6TiSCH node across a 6TiSCH multihop path.

3.2. Complex Tracks

A Complex Track is designed as a directed acyclic graph from a source 6TiSCH node towards a destination 6TiSCH node to support multi-path forwarding, as introduced in 6TiSCH Architecture [I-D.ietf-6tisch-architecture]. By employing DetNet [I-D.ietf-detnet-architecture] Packet Replication and Elimination (PRE) functions, several paths may be computed, and these paths may be more or less independent. For example, a complex Track may branch off and rejoin over non-congruent paths (branches).

In the following Section, we will detail Deterministic Networks PRE techniques.

4. Packet Replication and Elimination principles

In a nutshell, PRE consists in establishing several paths in a network to provide redundancy and parallel transmissions to bound the delay to traverse the network. Optionally, promiscuous listening between paths is possible, such that the nodes on one path may overhear transmissions along the other path. Considering the scenario depicted in Figure 1, many different paths are possible for

S to reach R. A simple way to take benefit from this topology could be to use the two independent paths via nodes A, C, E and via B, D, F. But more complex paths are possible by interleaving transmissions from one level of the path to the upper level.

The 6TiSCH PRE may also take advantage of the shared properties of the medium to compensate for the potential loss that is incurred with radio transmissions. For instance, when the source sends to A, B may listen and get a second chance to receive the frame without an additional transmission. Note that B would not have to listen if it already received that particular frame at an earlier timeslot.

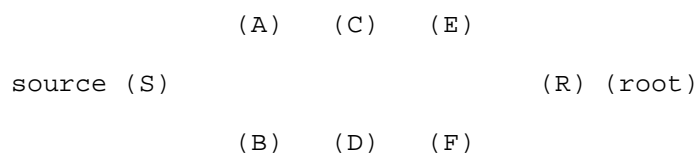


Figure 1: A Typical Ladder Shape with Two Parallel Paths Toward the Destination

PRE model can be implemented in both centralized and distributed scheduling approach. In the centralized approach, a Path Computation Element (PCE) scheduler calculates the routes and schedules the communication among the nodes along a circuit such as a Label switched path. In the distributed approach, each node selects its route to the destination, typically using a source routing header. In both cases, a default parent and alternate parent(s) should be selected to set up complex tracks.

In the following Subsections, detailed description of all required operations defined by PRE, namely, Alternative Path Selection, Packet Replication, Packet Elimination and Promiscuous Overhearing, will be described.

4.1. Packet Replication

The objective of PRE is to provide deterministic networking properties, with high reliability and bounded latency. To achieve this goal, determinism in every hop of the forwarding paths MUST be guaranteed. By employing Packet Replication procedure, each node transmits (i.e., replicates) each data packet to both its Default Parent (DP) and Alternative Parent (AP). To do so, each node (i.e., source and intermediate 6TiSCH nodes) transmits the data packet twice in unicast to each parent. For instance, in Figure 2, the source 6TiSCH node S is transmitting the packet to both parents, nodes A and

B, in two different timeslots within the same TSCH slotframe. Thus, the packet eventually obtains parallel paths to the destination.

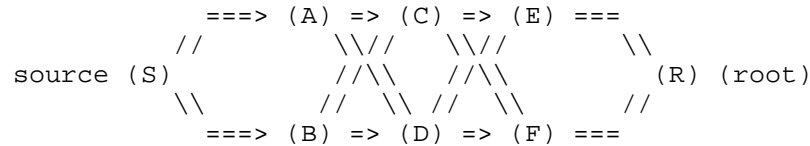


Figure 2: Packet Replication: S transmits twice the same data packet, to its DP (A) and to its AP (B).

4.2. Packet Elimination

The replication operation increases the traffic load in the network, due to packet duplications. Thus, Packet Elimination operation should be applied at each RPL DODAG level to reduce the unnecessary traffic. To this aim, once a node receives the first copy of a data packet, it discards the following copies. Because the first copy that reaches a node is the one that counts, it is the only copy that will be forwarded upward. Then, once a node performed the Packet Elimination operation, it will proceed with Packet Replication operation to forward the packet toward the RPL DODAG Root.

4.3. Promiscuous Overhearing

Considering that the wireless medium is broadcast by nature, any neighbor of a transmitter may overhear a transmission. By employing the Promiscuous Overhearing operation, DP and AP eventually have more chances to receive the data packets. In Figure 3, when node A is transmitting to its DP (node C), the AP (node D) and its Sibling (node B) may decode this data packet as well. As a result, by employing correlated paths, a node may have multiple opportunities to receive a given data packet. This feature not only enhances the end-to-end reliability but also it reduces the end-to-end delay.

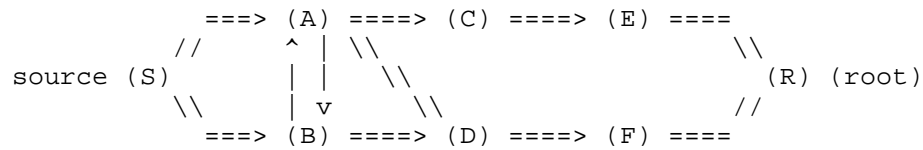


Figure 3: Unicast to DP with Overhearing: by employing Promiscuous Overhearing, DP, AP and the Sibling nodes have more opportunities to receive the same data packet.

5. Requirements

5.1. Requirements Related to Alternative Parent Selection

To perform the Replication procedure, it is necessary to define the Alternative Parent(s) and, consequently, the path to the destination 6TiSCH node, for each node in the 6TiSCH network. An AP can be selected in many different ways, and is dependent on the implementation.

Related requirements are:

Req1.1: The routing protocol SHOULD be extended to allow for each 6TiSCH node to select AP(s) in addition to DP. Thus, packet duplication (i.e., replication) to multiple parents could be possible.

Req1.2: Considering that the Replication procedure significantly increases the traffic in a network, when proposing solutions for Alternative Parent Selection, it should be efficient enough to mitigate the potential uncontrolled packet duplications.

Req1.3: The topology SHOULD be defined when proposing solutions for Alternative Parent Selection. For instance, the ladder topology should be defined explicitly e.g., number of parallel paths, density.

5.2. Requirements Related to Propagated Information

To select an Alternative Parent, 6TiSCH nodes MUST be aware of their grandparent node sets. Thus, it is necessary nodes to propagate such information to their neighbors. RPL [RFC6550] defines DODAG Information Object (DIO) Control Message to allow nodes to propagate information about themselves to potential children. In Figure 4, DIO control message with a DAG Metric Container option is illustrated. However, RPL [RFC6550], does not indicates how to propagate parent set related information.

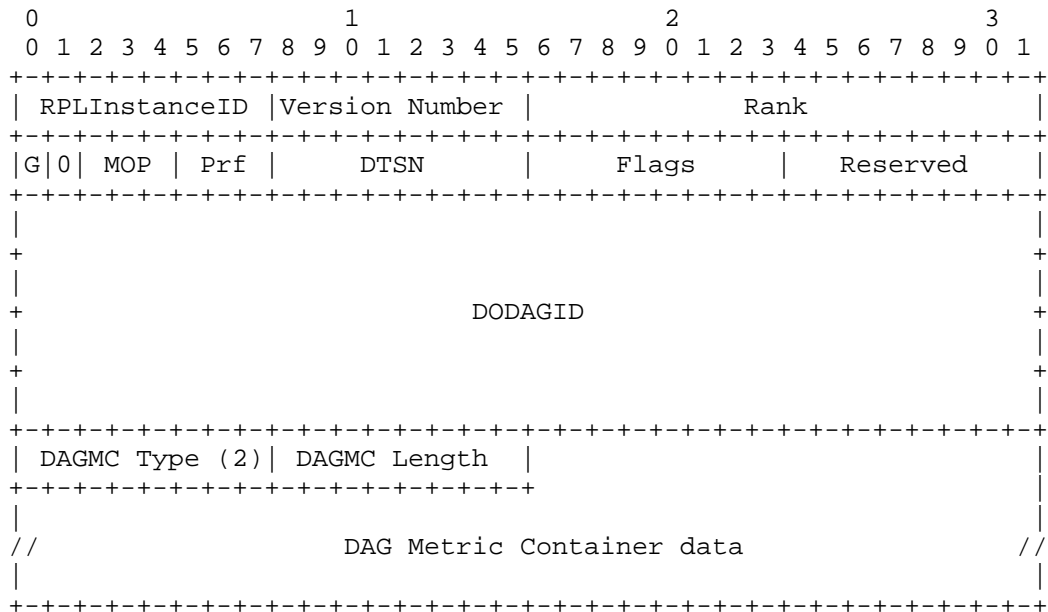


Figure 4: Example DIO Message with a DAG Metric Container option

Related requirements are:

Req2.1: DIO control messages can include multiple options. DAG Metric Container option [RFC6551] is structurally suitable for transferring parent node set information. Therefore, to enable PRE, nodes MUST broadcast their parent node set to their potential children through the extended DIO control message. For instance, "RPL DAG Metric Container (MC) Node State and Attribute (NSA) object type extension" [I-D.koutsiamanis-roll-nsa-extension] focuses on extending the DAG Metric Container [RFC6551] by defining new type-length-value (TLV), entitled Parent Node Set (PNS) which CAN be carried in the Node State and Attribute (NSA) object.

5.3. Requirements Related to Cell Reservation

As stated previously, to further increase the 6TiSCH network reliability and to achieve deterministic packet deliveries at the destination 6TiSCH node, Promiscuous Overhearing can be considered.

As it is described in BCP 210 [RFC8180], in TSCH mode, the data frames are transmitted in unicast mode and are acknowledged by the receiving neighbor. To perform the promiscuous overhearing procedure, there SHOULD be an option for the transmitted frames,

i.e., in unicast, to be overheard by the potential neighborhood 6TiSCH node.

Related requirements are:

Req3.1: The destination address filtering is performed at the MAC layer. According to IEEE std. 802.15.4 [IEEE802154-2015], a node receiving a packet with a destination address different than its own and different to 0xFF discards the packet. Thus, IEEE std. 802.15.4 implementation SHOULD bypass this filtering either by configuration forcing to accept such the receiving frame or by using anycast/multicast address as destination.

Req3.2: The 6top Protocol [I-D.ietf-6tisch-6top-protocol] SHOULD be extended to possibly allow a cell reservation with two receivers, i.e., DP and AP. Considering that each frame may be transmitted twice in unicast to each parent, then depending the transmission, either DP will acknowledge the frame or AP will.

Req3.3: Next, to request the overhearing cells, the 6P ADD Request Format SHOULD be transmitted either twice to each parent, i.e., DP and AP, or once in multicast to both parents. This procedure SHOULD be considered in 6top Protocol [I-D.ietf-6tisch-6top-protocol] specification.

5.4. Requirements Related to Cells without ACKs

As stated in BCP 210 [RFC8180], each data frame is acknowledged by the receiving 6TiSCH node. However, by employing promiscuous overhearing operation, particular attention should be given to who will acknowledge a transmission, i.e., the DP, and / or one of the AP(s)

Related requirements are:

Req4.1: To avoid the ACK collision, the TSCH Schedule as per BCP 210 [RFC8180], only the destination node of a packet MUST acknowledge the data packet.

Req4.2: The overhearing node can be configured with the timeslot set to shared, thus, there will be no acknowledgement from it. However, there is the security issue that needs to be considered. Since, the overhearing case imply that it is not possible to have per-pair keying, thus, there MUST be a key that the overhearing node will be aware of. Hence, Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-architecture] specification should consider such scenario.

Req4.3: Optionally, to achieve further consistency the overheard transmission need be acknowledged by both parents, i.e., DP and AP. To do so, MAC layer operation MUST be extended accordingly.

5.5. Requirements Related to Packet Elimination

By employing packet replication operation, the 6TiSCH network expects to perform the packet elimination operation along a complex Track to bound the number of the duplicated packets, i.e., the unnecessary traffic.

Related requirements are:

Req5.1: As per 6TiSCH Architecture [I-D.ietf-6tisch-architecture], 6TiSCH has no position about how the sequence numbers would be tagged in the packet. However, it comes with Tagging Packets for Flow Identification. More specifically, a 6TiSCH network expects that timeslots corresponding to copies of a same frame along a complex Track are correlated by configuration and, thus, does not need to process the sequence numbers.

6. Security Considerations

TODO.

7. IANA Considerations

This document has no IANA considerations.

8. References

8.1. Informative references

[I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer Protocol (6P)", draft-ietf-6tisch-6top-protocol-12 (work in progress), June 2018.

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-14 (work in progress), April 2018.

[I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-06 (work in progress), May 2018.

[I-D.ietf-detnet-architecture]

Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", draft-ietf-
detnet-architecture-06 (work in progress), June 2018.

[I-D.koutsiamanis-roll-nsa-extension]

Koutsiamanis, R., Papadopoulos, G., Montavont, N., and P.
Thubert, "RPL DAG Metric Container Node State and
Attribute object type extension", draft-koutsiamanis-roll-
nsa-extension-02 (work in progress), July 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J.,
Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur,
JP., and R. Alexander, "RPL: IPv6 Routing Protocol for
Low-Power and Lossy Networks", RFC 6550,
DOI 10.17487/RFC6550, March 2012,
<<https://www.rfc-editor.org/info/rfc6550>>.

[RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N.,
and D. Barthel, "Routing Metrics Used for Path Calculation
in Low-Power and Lossy Networks", RFC 6551,
DOI 10.17487/RFC6551, March 2012,
<<https://www.rfc-editor.org/info/rfc6551>>.

[RFC8180] Vilajosana, X., Ed., Pister, K., and T. Watteyne, "Minimal
IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH)
Configuration", BCP 210, RFC 8180, DOI 10.17487/RFC8180,
May 2017, <<https://www.rfc-editor.org/info/rfc8180>>.

8.2. Other Informative References

[IEEE802154-2015]

IEEE standard for Information Technology, "IEEE standard
for Information Technology, "IEEE Std 802.15.4-2015
Standard for Low-Rate Wireless Personal Area Networks
(WPANs)", December 2015".

Authors' Addresses

Georgios Papadopoulos (editor)
IMT Atlantique
Office B00 - 102A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 04
Email: georgios.papadopoulos@imt-atlantique.fr

Nicolas Montavont
IMT Atlantique
Office B00 - 106A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 23
Email: nicolas.montavont@imt-atlantique.fr

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com