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.

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.

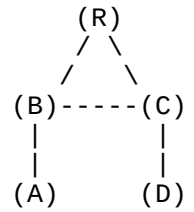


Figure 1: A simple 6TiSCH network.

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:

- 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.
- 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.
- 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.

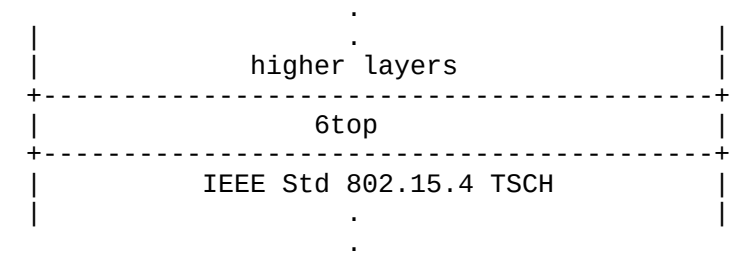


Figure 2: The 6top sublayer in the protocol stack.

The roles of the 6top sublayer are to:

- Terminate the 6top Protocol (6P), which allows neighbor nodes to communicate to add/delete cells to one another.
- 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":

- a soft cell can be read, added, deleted or updated by 6top.
- 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:

- 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.
- 6P allocates cells from Slotframe 1. In Figure 3, Slotframe 1 is 10 slots long, but it can be shorter or longer.

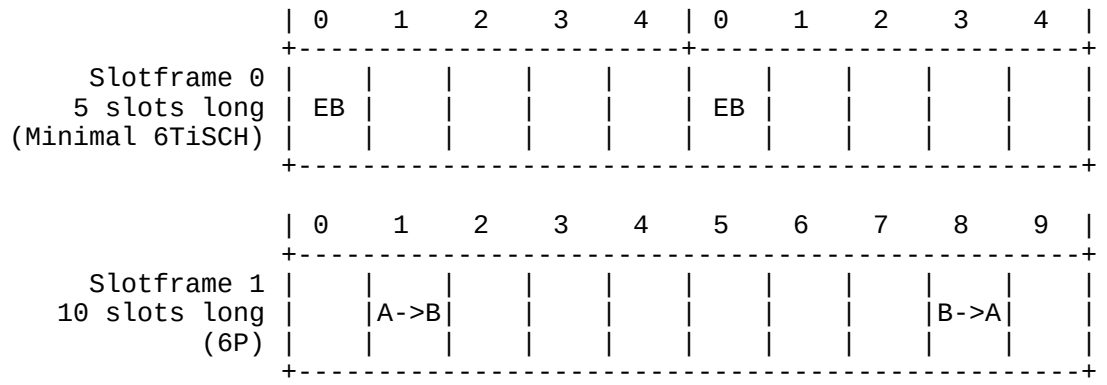


Figure 3: 2-slotframe structure when using 6P alongside the Minimal 6TiSCH Configuration.

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. 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.

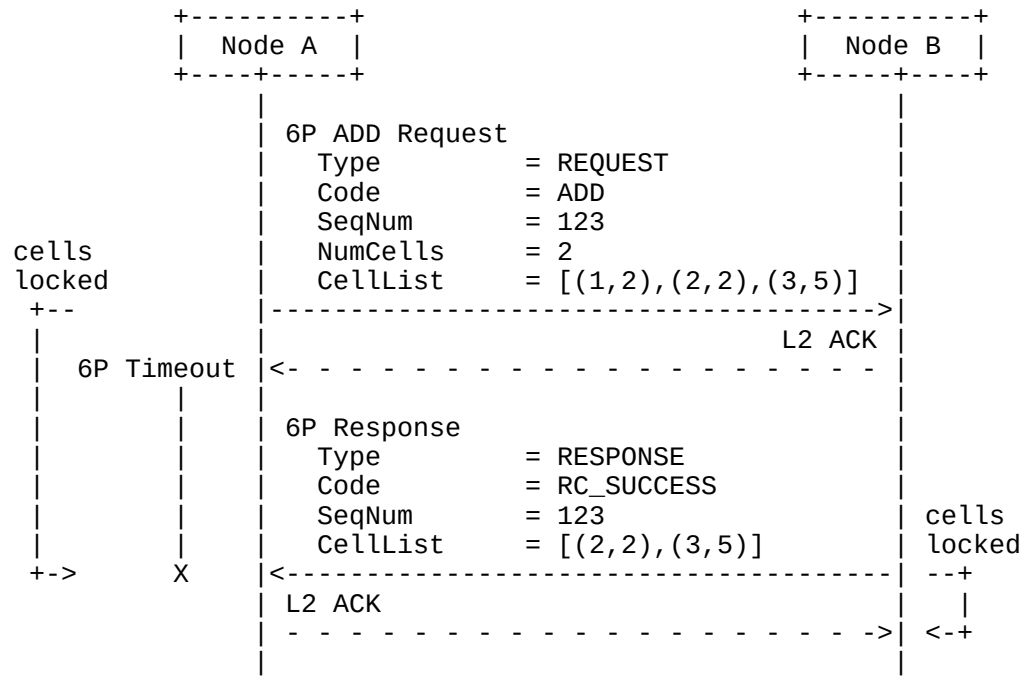


Figure 4: An example 2-step 6P Transaction.

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.

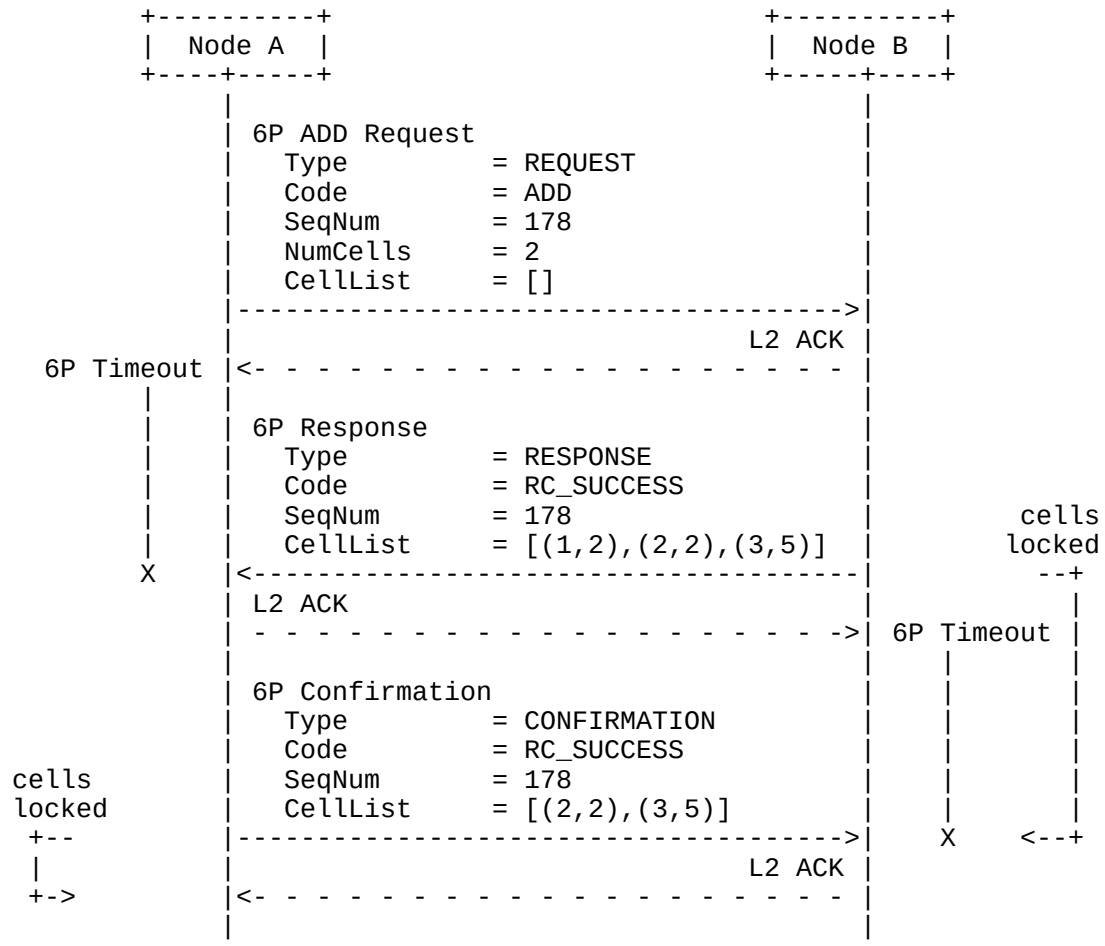


Figure 5: An example 3-step 6P Transaction.

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.

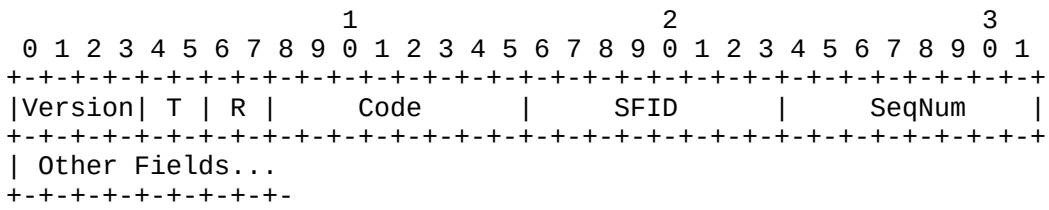


Figure 6: Generic 6P Message Format.

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.

- In the 6P ADD request, the 6P CellOptions bitmap is used to specify what type of cell to add.
- In the 6P DELETE request, the 6P CellOptions bitmap is used to specify what type of cell to delete.
- In the 6P RELOCATE request, the 6P CellOptions bitmap is used to specify what type of cell to relocate.
- 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:

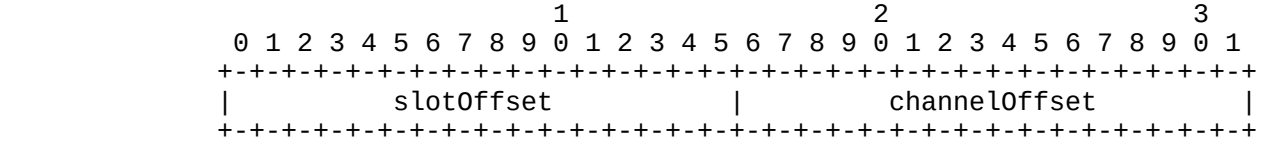


Figure 9: 6P Cell Format.

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.

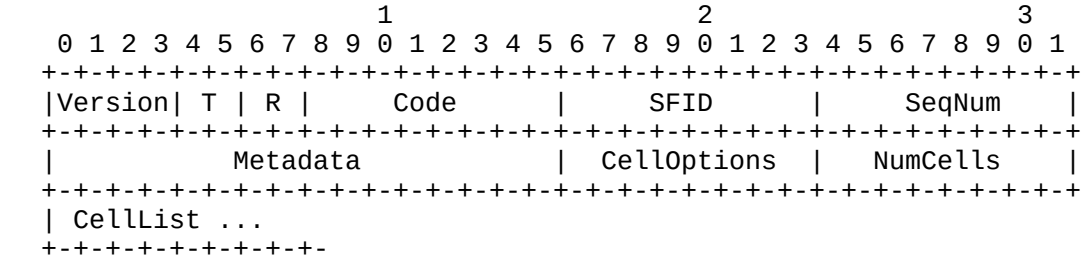


Figure 10: 6P ADD Request Format.

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.

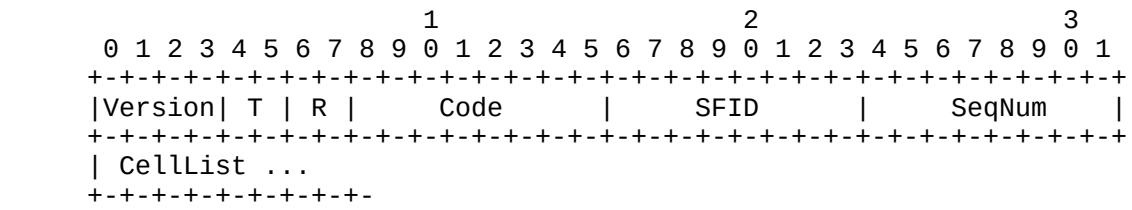


Figure 11: 6P ADD Response and Confirmation Formats.

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.

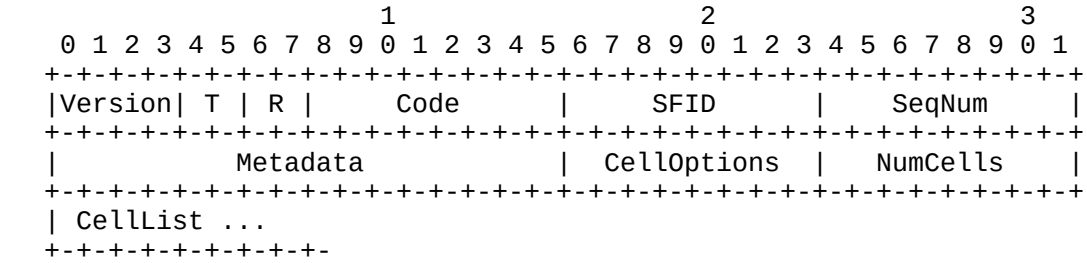


Figure 12: 6P DELETE Request Format.

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 are 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.

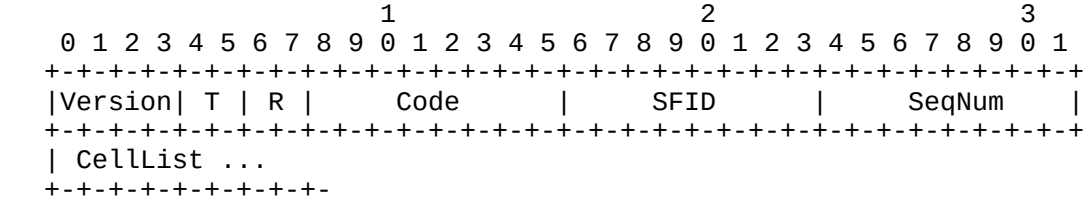


Figure 13: 6P DELETE Response and Confirmation Formats.

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

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

- The nodes delete the cells they agree upon rather than adding them.
- 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.
- 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.

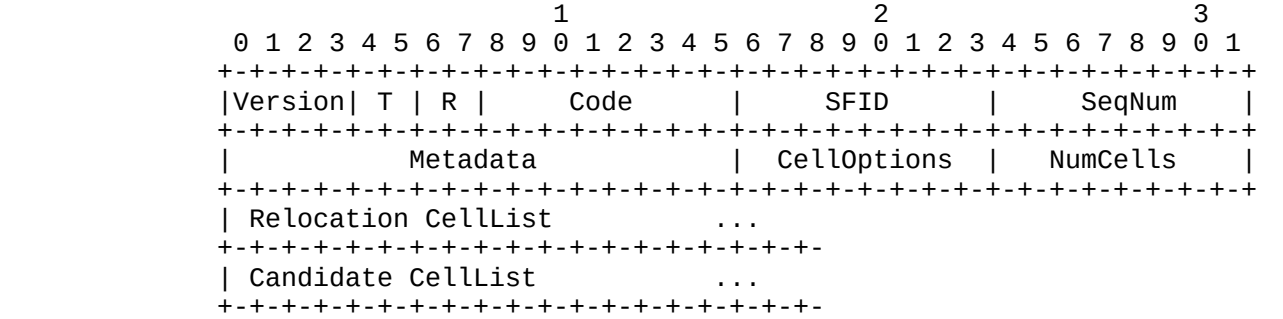


Figure 14: 6P RELOCATE Request Format.

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.

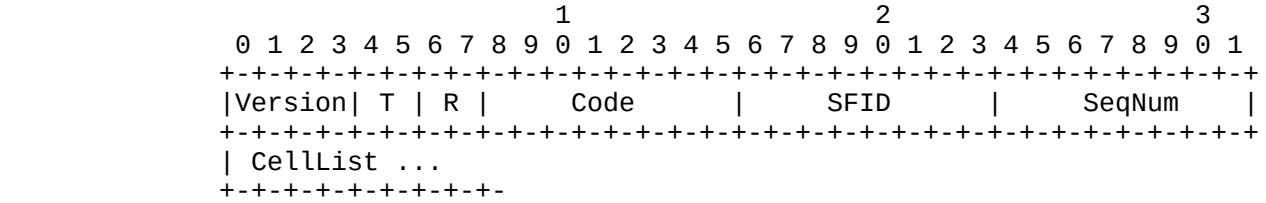


Figure 15: 6P RELOCATE Response and Confirmation Formats.

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 < 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.

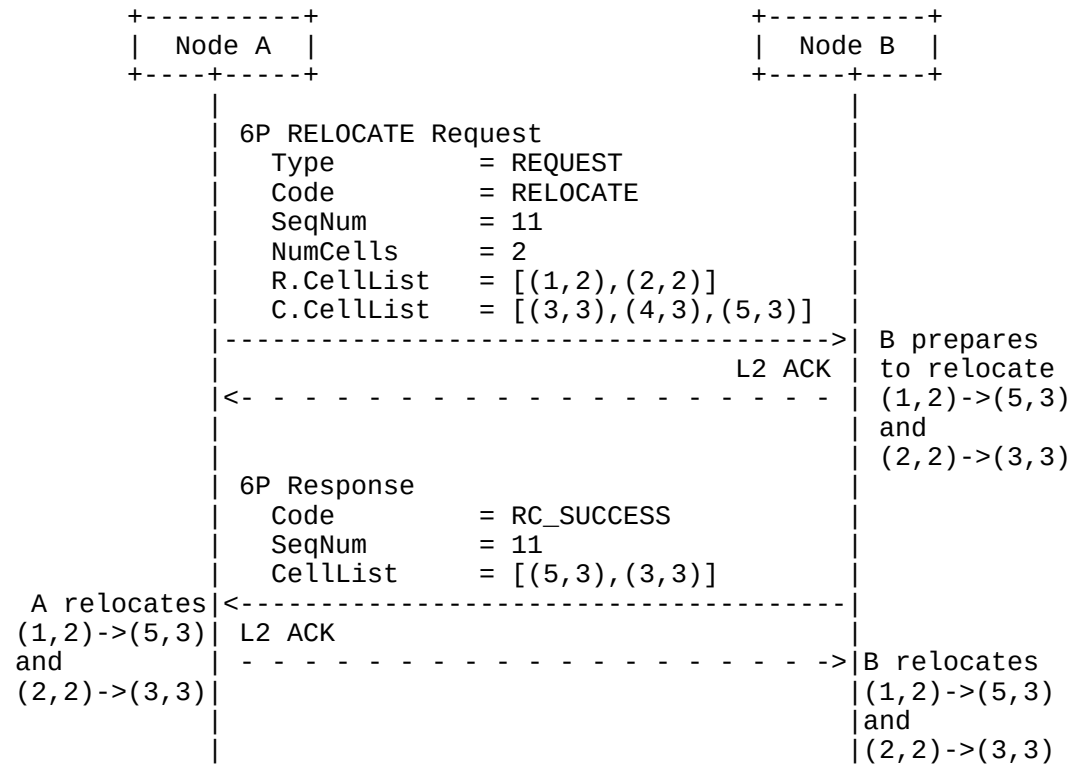


Figure 16: Example of a successful 2-step 6P RELOCATION Transaction.

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

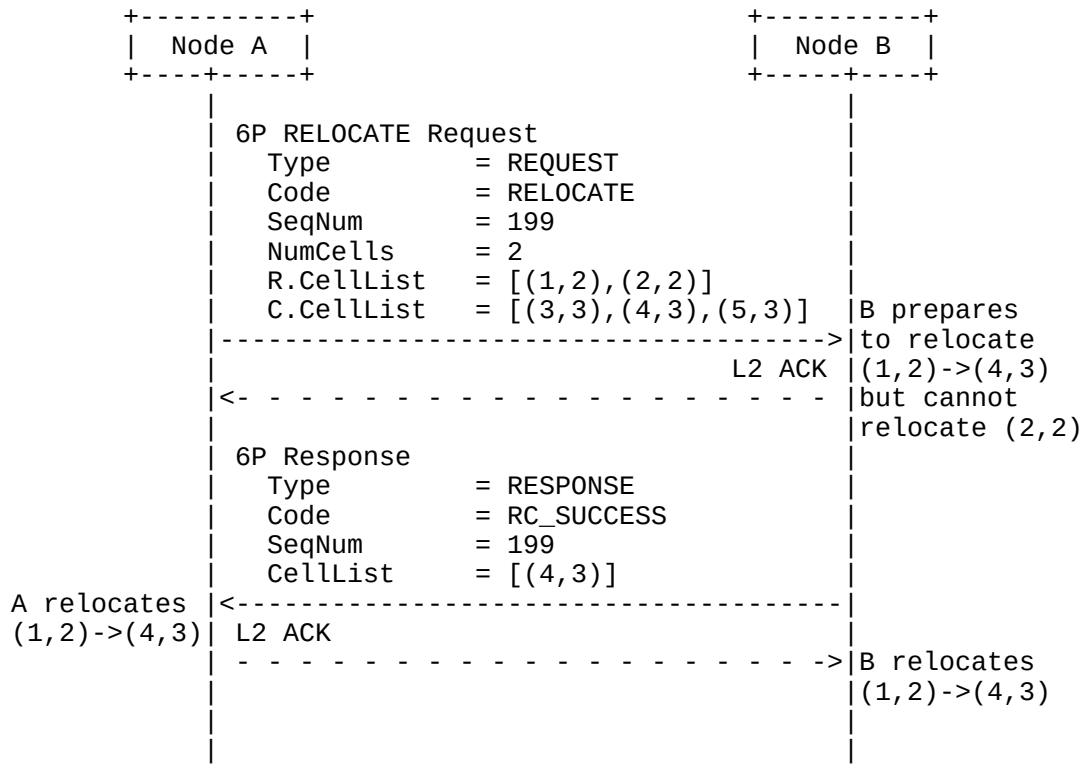


Figure 17: Example of a partially successful 2-step 6P RELOCATION Transaction.

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

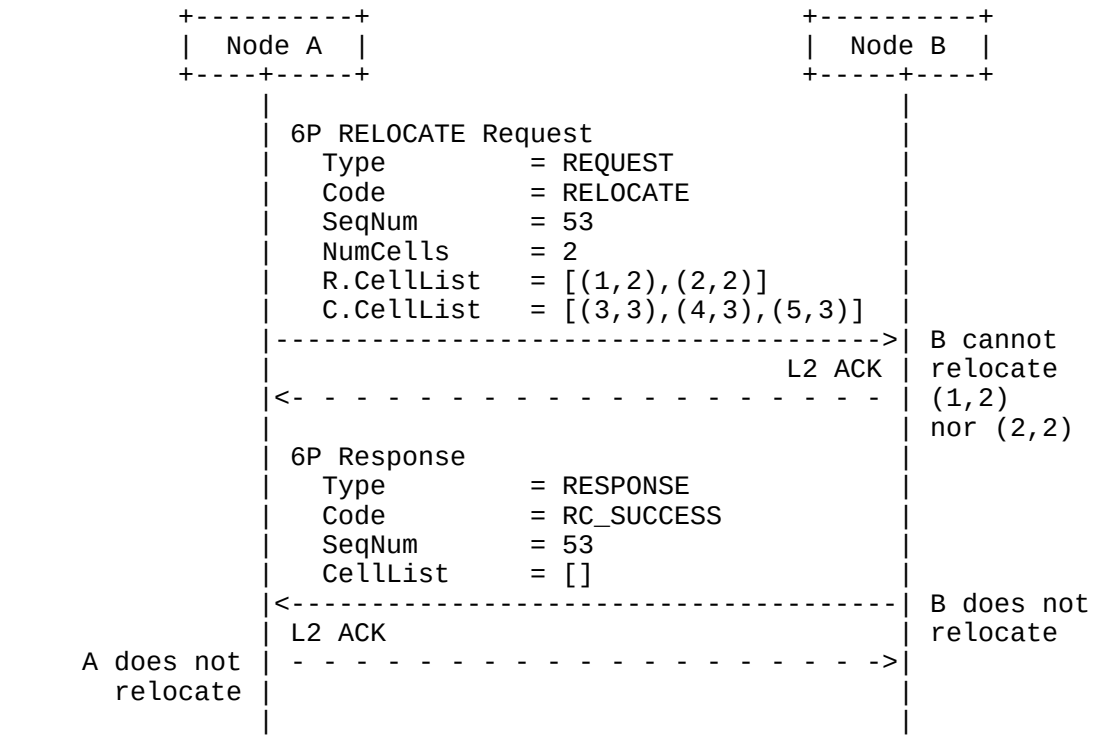


Figure 18: Failed 2-step 6P RELOCATION Transaction Example.

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

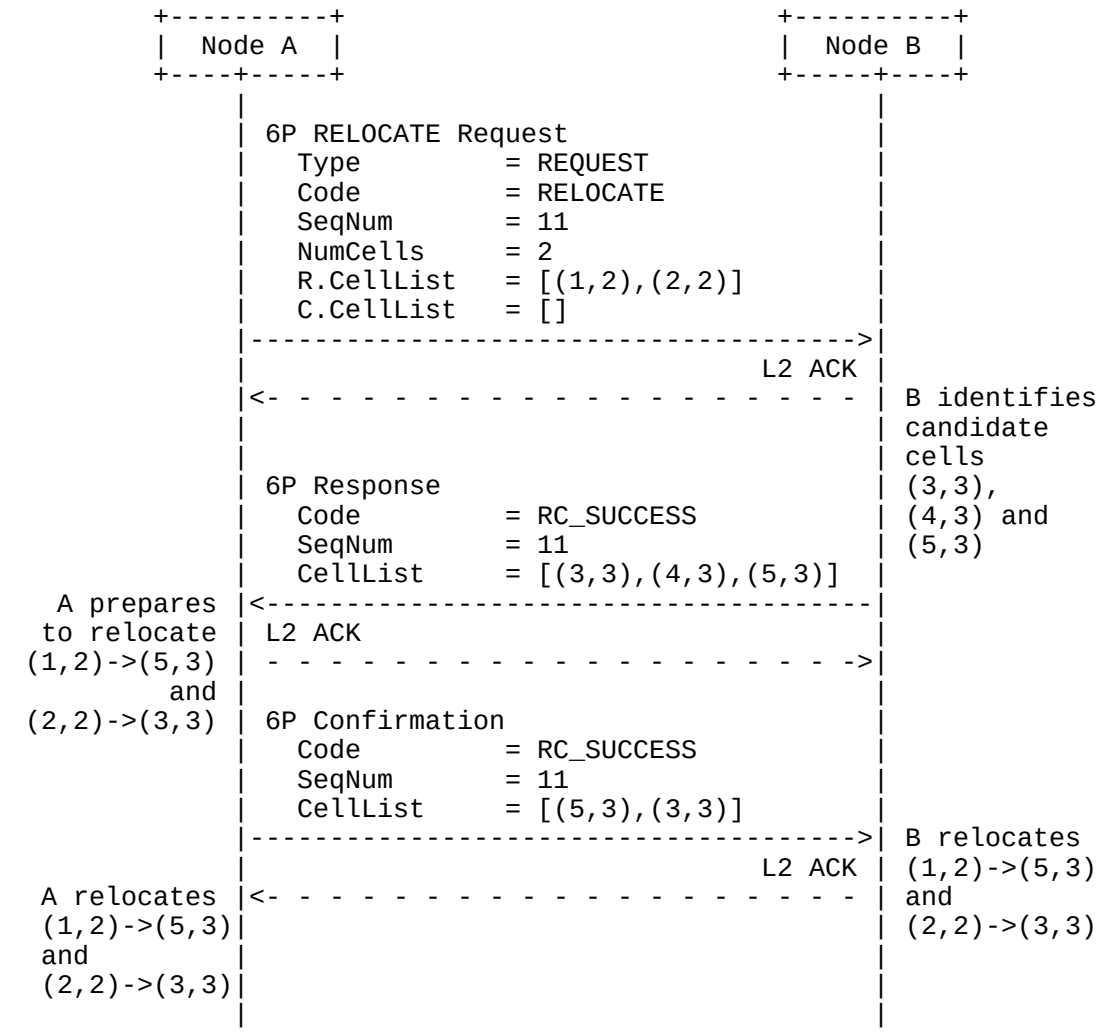


Figure 19: Example of a successful 3-step 6P RELOCATION Transaction.

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.

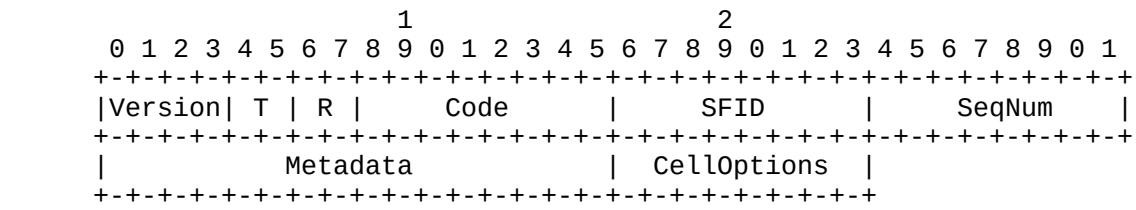


Figure 20: 6P COUNT Request Format.

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.

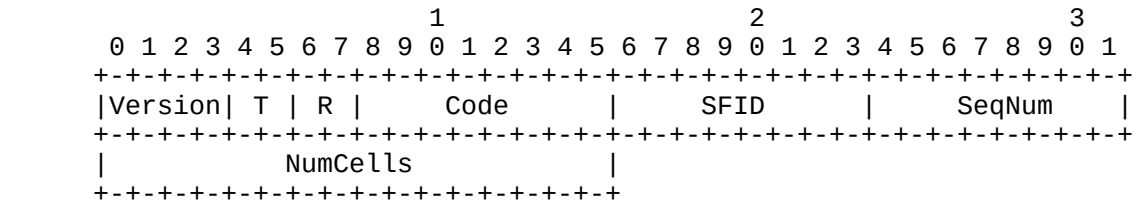


Figure 21: 6P COUNT Response Format.

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.

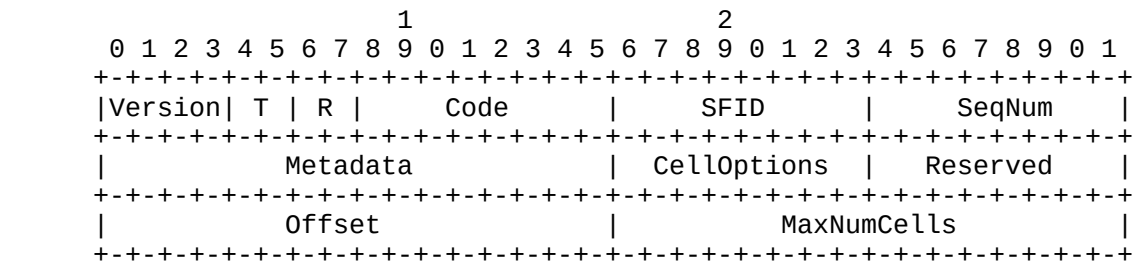


Figure 22: 6P LIST Request Format.

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.

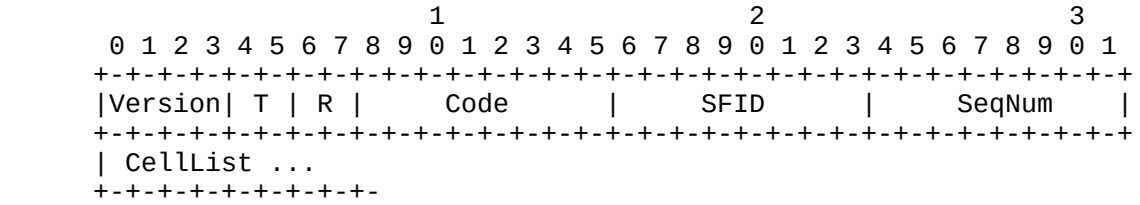


Figure 23: 6P LIST Response Format.

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.

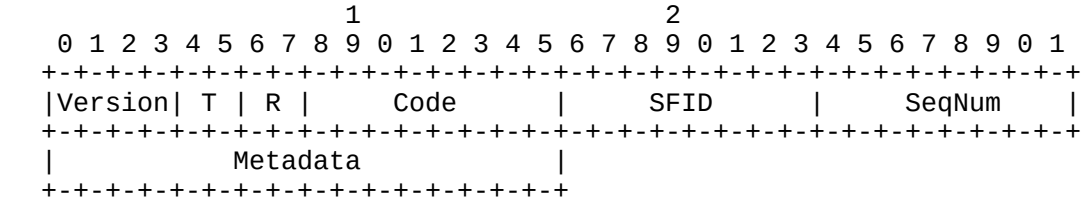


Figure 24: 6P CLEAR Request Format.

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.

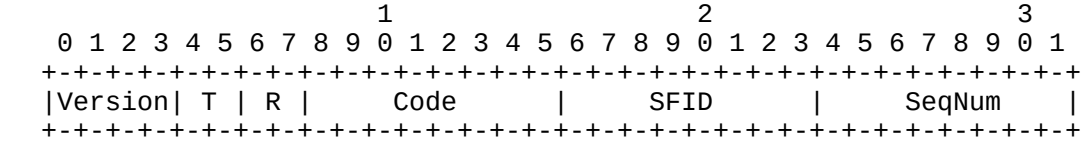


Figure 25: 6P CLEAR Response Format.

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.

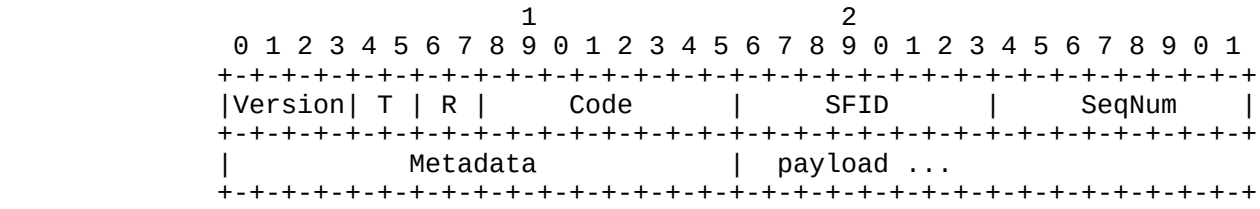


Figure 26: 6P SIGNAL Request Format.

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.

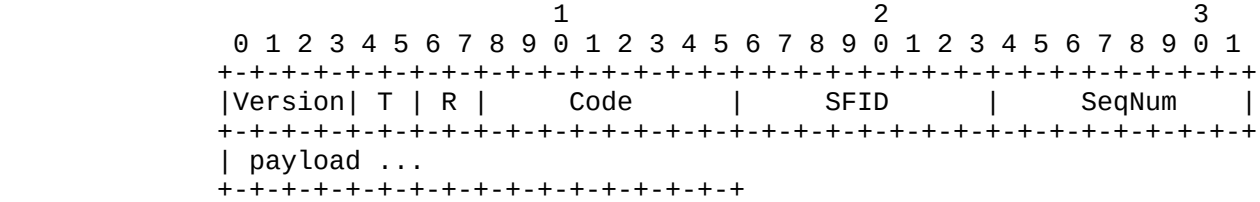


Figure 27: 6P SIGNAL Response Format.

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.

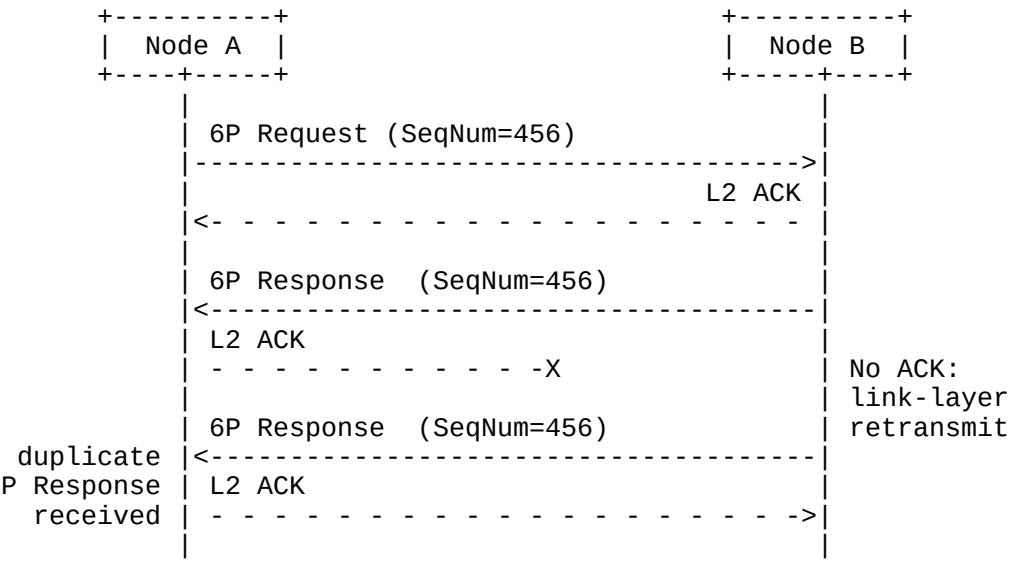


Figure 29: Example duplicate 6P message.

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.

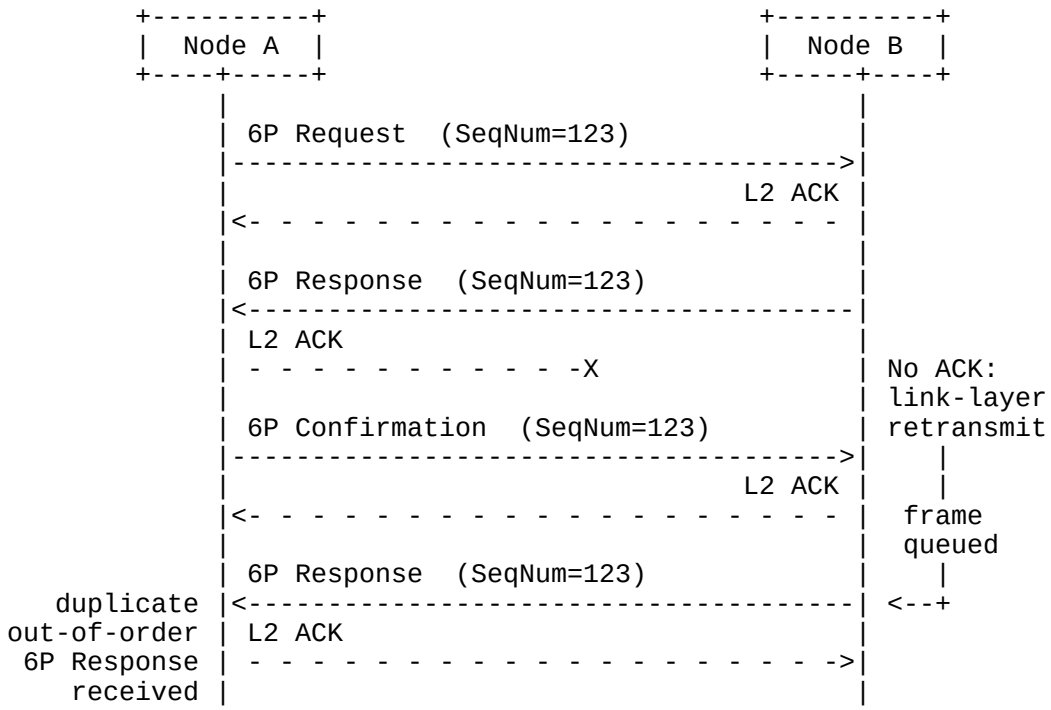


Figure 30: Example out-of-order duplicate 6P message.

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.

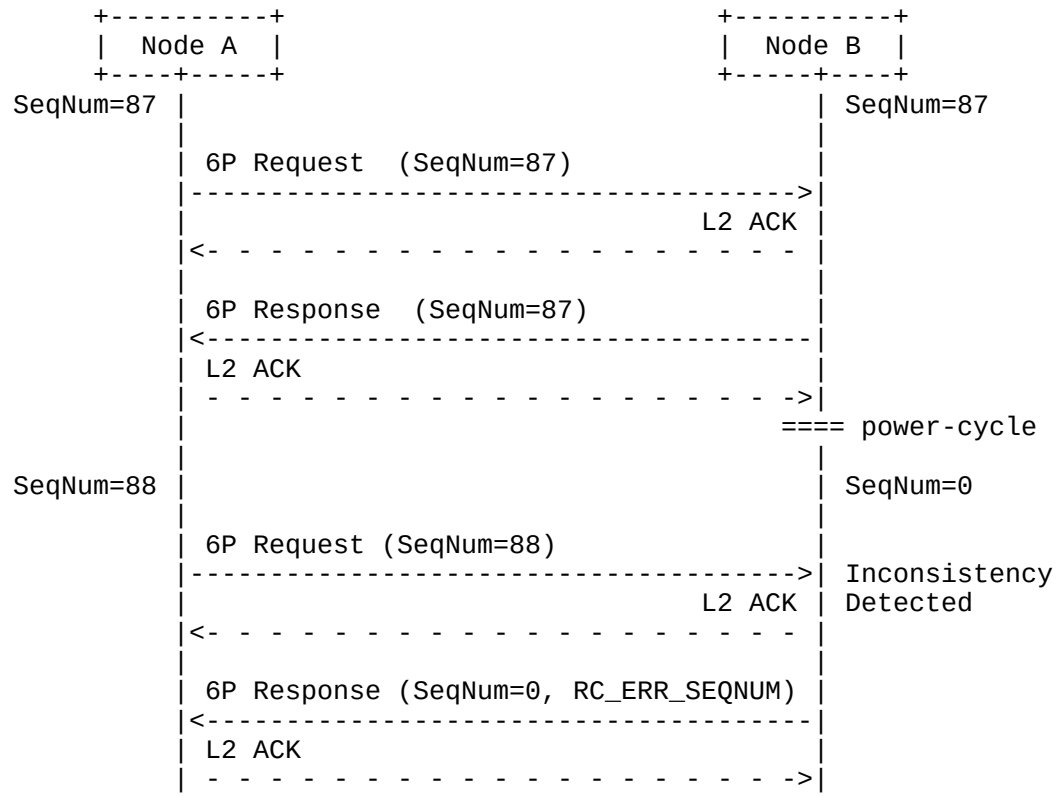


Figure 31: Example of inconsistency because of node B reset. Detected by node B

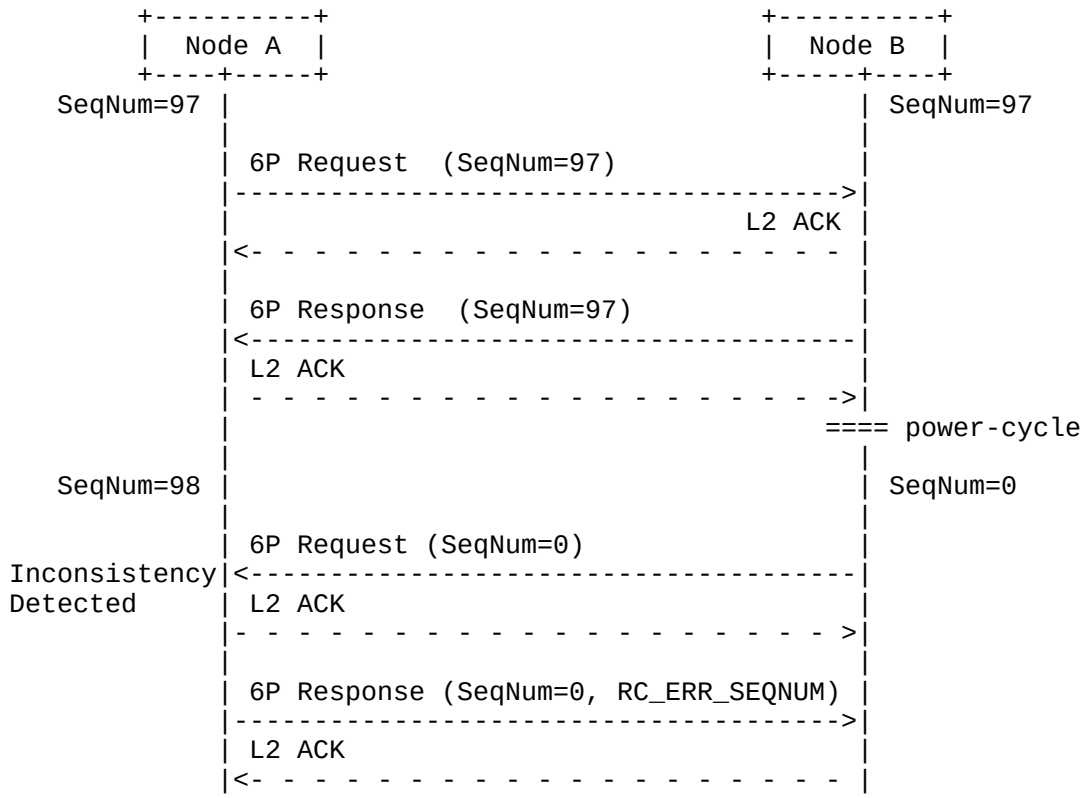


Figure 32: Example of inconsistency because node B resets. Detected by node A

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.

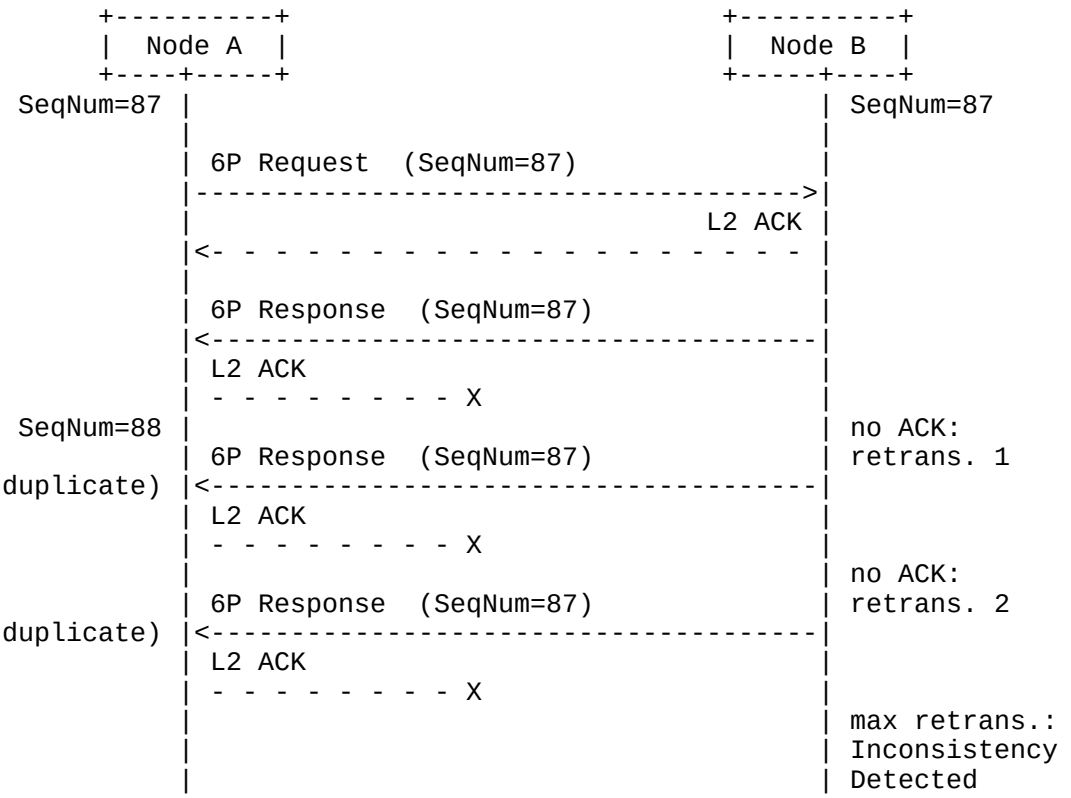


Figure 33: Example inconsistency because of maximum link-layer retransmissions (here 2).

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

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

- MUST specify what to do after an error has occurred (either the node sent a 6P Response with an error code, or received one).
- 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.

- SHOULD clearly state the application domain the SF is created for.
- SHOULD contain examples which highlight normal and error scenarios.
- SHOULD contain a list of current implementations, at least during the I-D state of the document, per [RFC6982].
- SHOULD contain a performance evaluation of the scheme, possibly through references to external documents.
- SHOULD define the format of the SIGNAL command payload and its use.

- MAY redefine the format of the CellList field.
- MAY redefine the format of the CellOptions field.
- 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:

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

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:

- "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.
- "TEMPORARY" appendices are there to capture current ongoing discussions, or the changelog of the document. These appendices will be removed in the final text.
- "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.
- The string "REMARK" is put before a remark (questions, suggestion, etc) from an author, editor of contributor. These are on-going discussions at the time to writing, NOT part of the final text.
- 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 absorbed and detected.

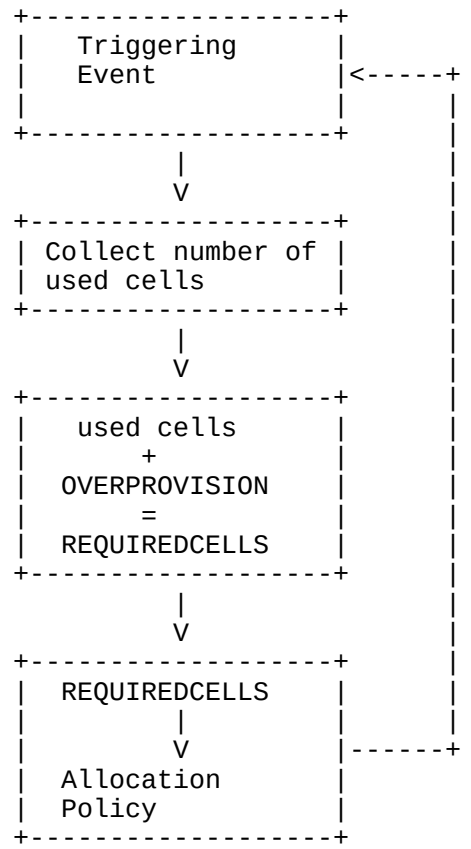


Figure 1: The SF0 Estimation Algorithm

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.

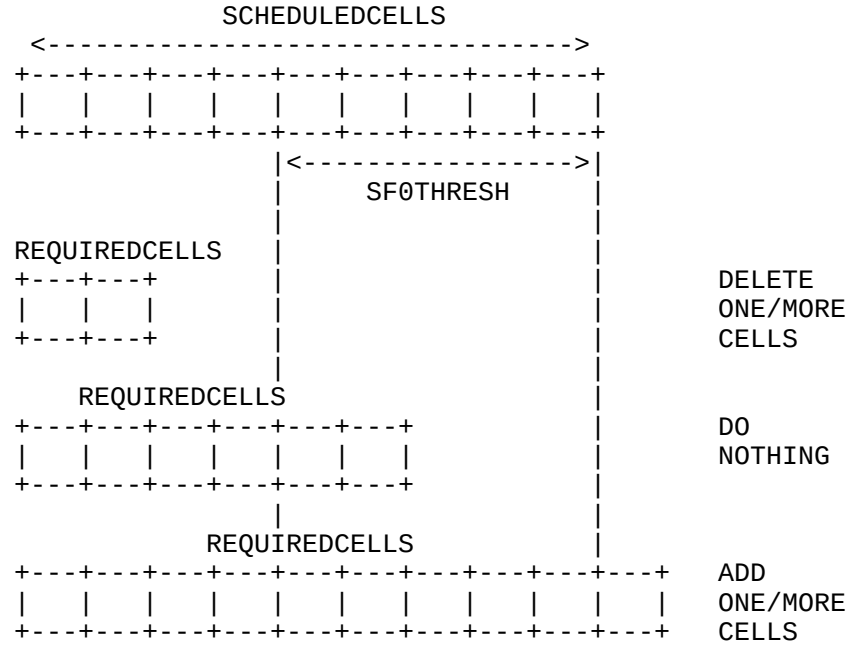


Figure 2: The SF0 Allocation Policy

1. If REQUIREDCELLS<(SCHEDULEDCELLS-SF0THRESH), delete one or more cells.
2. If (SCHEDULEDCELLS-SF0THRESH)<=REQUIREDCELLS<=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.

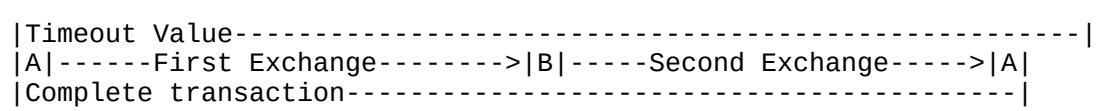


Figure 3: Example Transaction where the timeout does not expire

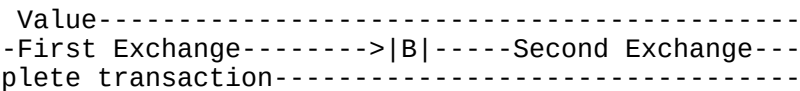


Figure 4: Example Transaction where the timeout expires

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

- IANA_6TiSCH_SFID_SF0

20. 6P Compliance

- MUST specify an identifier for that SF. OK
- MUST specify the rule for a node to decide when to add/delete one or more cells to a neighbor. OK
- MUST specify the rule for a Transaction source to select cells to add to the CellList field in the 6P ADD Request. OK
- MUST specify the rule for a Transaction destination to select cells from CellList to add to its schedule. OK
- MUST specify a value for the 6P Timeout, or a rule/equation to calculate it. OK
- MUST specify a meaning for the "Metadata" field in the 6P ADD Request. OK
- MUST specify the behavior of a node when it boots. OK
- 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
- MUST specify the list of statistics to gather. An example statistic if 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
- SHOULD clearly state the application domain the SF is created for. OK
- SHOULD contain examples which highlight normal and error scenarios.
- SHOULD contain a list of current implementations, at least during the I-D state of the document, per [RFC6982].
- SHOULD contain a performance evaluation of the scheme, possibly through references to external documents.
- 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

- draft-ietf-6tisch-6top-sf0-02
  - Editorial changes (figs, typos, ...)
- 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: April 23, 2017

M. Richardson
Sandelman Software Works

October 20, 2016

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

Abstract

Charter: The WG will continue working on securing the join process and making that fit within the constraints of high latency, low throughput and small frame sizes that characterize IEEE802.15.4 TSCH.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2017.

Table of Contents

1. Introduction 3
1.1. Terminology 3
1.2. Credentials 4
1.2.1. One-Touch Assumptions 4
1.2.2. Factory provided credentials (if any) 5
1.2.3. Credentials to be introduced 5
1.3. Network Assumptions 5
1.3.1. Security above and below IP 6
1.3.2. Join network assumptions 7
1.3.3. Number and cost of round trips 7
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. Diagram of Join Process 7
2.2. Description of Pledge States in Join Process 8
2.2.1. (1) Layer-2 Enhanced Beacon 8
2.2.2. (1B) Layer-3 Router Advertisement 8
2.2.3. (2) Pledge sends (unicast) Neighbor Solicitation to Join Assistant 8
2.2.4. (3) Join Assistant sends Query to Registar 9
2.2.5. (4) Join Assistant receives Acceptance response from Registrar 9
2.2.6. (5) Pledge receives (unicast) Neighbor Advertisement from join assistant 9
2.3. Join process state machine for pledge 10
2.4. Description of Join Assistant States in Join Process . . 11
2.4.1. Processing of Insecure Packets 12
3. Join Assistant to Registrar protocol 13
3.1. Discovery of Registrar 13
3.2. Notification of a new pledge to Registar 14
3.3. Passing of traffic from Pledge to Registar 14
4. Privacy Considerations 15
4.1. Privacy Considerations for Production network 15
4.2. Privacy Considerations for New Pledges 15
4.2.1. EUI-64 derived address for join time IID 16
4.3. Privacy Considerations for Join Assistant 16
5. Security Considerations 17
6. IANA Considerations 17
7. Protocol Definition 17
8. References 17
8.1. Normative References 17
8.2. Informative References 19
Appendix A. appendix 20
Author's Address 20

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 undesireable 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.

In other installations (such as some factory/industrial settings, and for some utilities), it is possible to pass devices through a "provisioning" room of some kind where the device in factory default state may be touched once (via a cable, or a push button, or by being placed in a faraday cage, etc.) such that the devices can be initialized in a way specific to that installation. The devices are then returned to inventory, and may be deployed at some future time. The node is not provisioned with the current keying material, as the network will need to be regularly rekeyed (even the algorithms may change!), so in the one-touch provisioning case, the goal is simply to introduce some elements into the new node (the "pledge") such that the enrollment process will take less energy and fewer network resources.

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 following terminology is repeated from [I-D.ietf-anima-bootstrapping-keyinfra] so as to have a common way to speak:

drop ship The physical distribution of equipment containing the "factory default" configuration to a final destination. In zero-touch scenarios there is no staging or pre-configuration during drop-ship.

imprint the process where a device obtains the cryptographic key material to identity and trust future interactions with a network. This term is taken from Konrad Lorenz's work in biology with new ducklings: during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother. An equivalent for a device is to obtain the fingerprint of the network's root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Securely imprinting is a primary focus of this document. [duckling] anticipates this use.

enrollment the process where a device presents key material to a network and acquires a network specific identity. For example when a certificate signing request is presented to a certification authority and a certificate is obtained in response.

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. This is definition 6, according to [pledge].

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-zerotouch].

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 manufacturer 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

In a one-touch scenario, devices would be 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: * JTAG interface * serial (craft) console interface * pushes of physical buttons simulatenous to network attachment * insecured 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 Coordination Entity, and the Pledge, and that the two devices can exchange some trusted bytes of information.

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.

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:

- a network key that is shared with all nodes and is used for unicast and multicast.
- a series of network keys that are shared (agreed to) between pairs of nodes (the per-peer key)
- a network key that is shared with all nodes (through a group key management system), and is used for multicast traffic only

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.

- to provide for secure connection with a Path Computation Element [RFC4655], or other LLC (see ({RFC7554}} section 3).
- 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

Perfert 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:

- a known PANID. The PANID 0xXXXX where XXXX is the assigned RFC# for this document is suggested.
- a minimal schedule with some Aloha time. This is usually in the same slotframe as the Extended Beacon, but a pledge MUST listen for an unencrypted Extended Beacon to so that it can synchronize.
- a known K1 key, as described in [I-D.ietf-6tisch-minimal] section 10, and used for reasons similar to [iec62591].

1.3.3. Number and cost of round trips

1.3.4. Size of packets, number of fragments

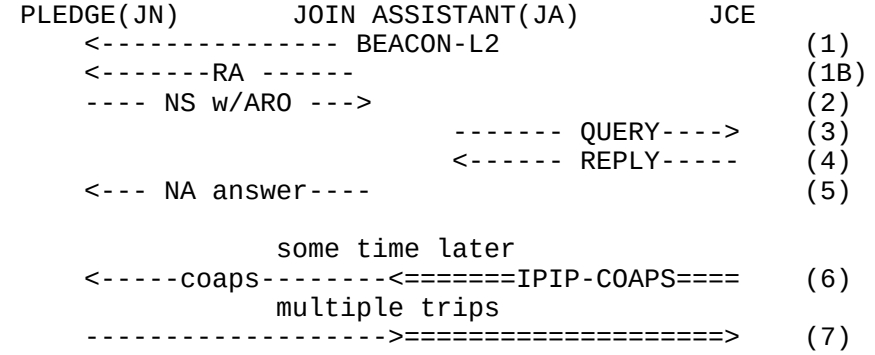
1.4. Target end-state for join process

2. Join Protocol

This section describes the interaction between a new pledge and the Join Assistant.

2.1. Diagram of Join Process

This time sequence diagram intentionally shows additional layer-2 and layer-3 interactions, in order to put the entire process into context.



2.2. Description of Pledge States in Join Process

2.2.1. (1) Layer-2 Enhanced Beacon

A new pledge must listen for a beacon for a period of at least 2s (HELP) on each of the 16 possible channels. The pledge SHOULD collect all beacons from heard on all channels before selecting a beacon to start with. If the pledge is unable to record all of the beacons that it hears due to limitations on volatile storage, then it should at least attempt to record the detail as to how to find that beacon again (channel, time sequence).

This search process entails having the pledge keep the receiver portion of it’s radio active for the entire period of time.

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 Beacon has been tried before, whether a Router Advertisement IE is present, any Network Identifier [I-D.richardson-6lo-ra-in-ie] seen, and the strength of the signal.

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

2.2.2. (1B) Layer-3 Router Advertisement

If [I-D.richardson-6lo-ra-in-ie] has not been used to embed a router advertisement in the Enhanced Beacon, then the pledge MUST wait to hear a Router Advertisement to know the layer-3 address of the adjacent router. These will be broadcast periodically during the ALOHA slot.

A pledge MAY timeout and send a layer-2 unicast Router Solicitation (to the layer-2 of the Enhanced Beacon) to the layer-3 all-routers address. A pledge MAY also take this timeout to mean that this router is unwilling to perform Join Assistant activities and the pledge should move on to another Enhanced Beacon.

2.2.3. (2) Pledge sends (unicast) Neighbor Solicitation to Join Assistant

This NS message is formed much like a Duplicate Address Detection (DAD) message described in [RFC6775] section-4.3: it is a solicitation by the pledge for it’s own address. [RFC6775] does not describe doing DAD for link-local address however, so this aspect is new.

The Join Assistant will not validate the uniqueness using the DAR/DAC mechanism, but will otherwise process the NS as per normal: populating neighbor cache entries. The Join Assistant will take extra care with expiring neighbor cache entries: unsecured NS should never push secured NCE entries out of the cache or overwrite them. There are two equivalent ways to do this:

1. marking the origin of the NCE and limiting unsecured ones to some portion of the entries;

2. by considering unsecured NS to be arriving from a different virtual interface (different if_index) than secured ones. NCEs from different interfaces SHOULD already not be mixed.

The pledge SHOULD NOT have configured a short Layer-2 address as it has no way to allocate a non-duplicate short address. It SHOULD have formed a standard 64-bit layer-3 link-local address using a built-in IID. This IID MUST be placed into the Address Resolution option (ARO) option in the Neighbor Soliciation, as it serves as the index by which the domain registrar will use to identify the device.

The IID MAY be related to the layer-2 address, but privacy considerations recommend that the IID SHOULD instead be a form a stable privacy address [RFC7217]. Whichever method is used MUST be decided at manufacturing time, as the IID is also repeated as the SerialNumber in the Manufacturer Installed Certificate (MIC), also referred to as the 802.1AR IDevID.

2.2.4. (3) Join Assistant sends Query to Registar

This step does not involve the pledge, and it is described in section (#jastates).

2.2.5. (4) Join Assistant receives Acceptance response from Registrar

This step does not involve the pledge, and it is described in section (#jastates).

2.2.6. (5) Pledge receives (unicast) Neighbor Advertisement from join assistant

This NA message is again identical to the Duplicate Address Detection mechanism described in [RFC6775]. The status field of the ARO is extended (see (#ianaconsiderations)) to include an additional status value ND_NS_JOIN_DECLINED.

The pledge, upon receipt of ND_NS_JOIN_DECLINED considers that this network is not an appropriate network to join, and SHOULD move on to

attempt other networks. The pledge MUST also realize that this NA message MAY have been forced, and it SHOULD attempt joining this network again at a future time, but MUST NOT repeatedly attempt to join the same network.

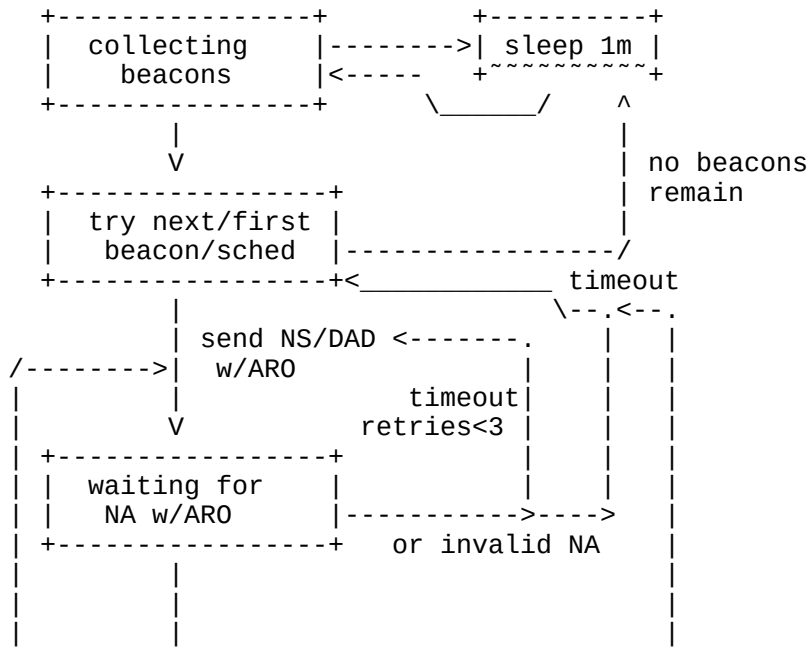
The pledge, upon receipt of a Neighbor Cache Full response SHOULD attempt to join using a different join assistant on the same network.

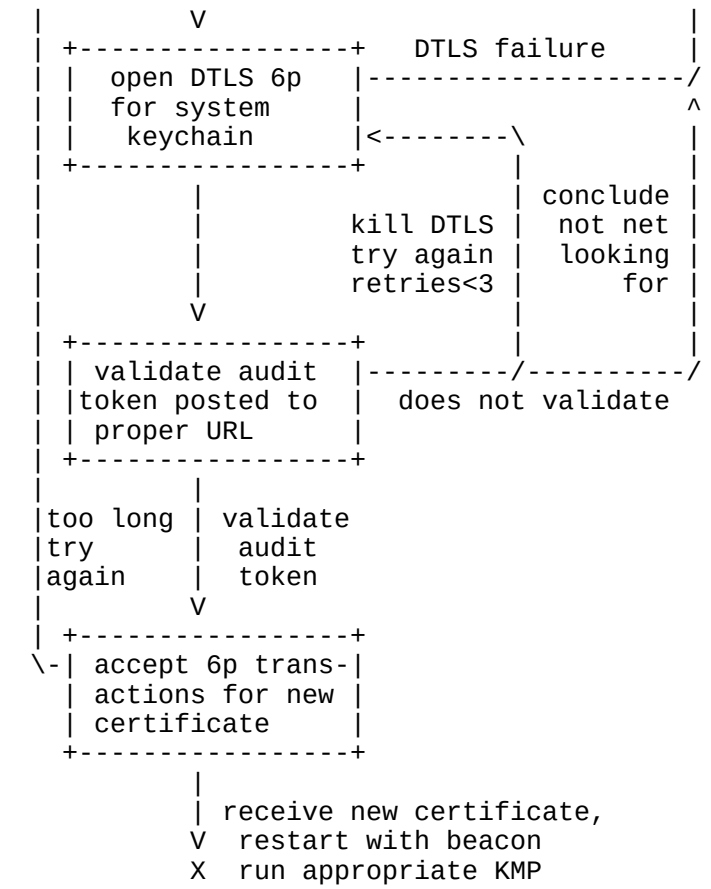
The pledge, upon receipt of a Duplicate Address response SHOULD attempt to join using a different join assistant on a different network, if it has such offers. If it has no such offers, it should wait at least NEIGHBOR-CACHE TIMEOUT, and then retry. This may be a sign of a Denial of Service attack, or it may be a non-malicious mis-configuration.

Upon receive of a successful NA, the pledge SHOULD consider that it is now in enrolled in a join queue. The pledge SHOULD resend Neighbor Soliciation (NUD) messages periodically as described in [RFC6775] to maintain the neighbor cache entry.

A pledge with other Join Assistant offers MAY abandon this Join Assistant after a period of XXX minutes and attempt to join using a different Join Assistant.

2.3. Join process state machine for pledge





2.4. Description of Join Assistant States in Join Process

The Join Assistant is a standard 6LR. It processes packets as described in [RFC6775], [I-D.ietf-6tisch-minimal] from secured (encrypted) sources.

In particular the maintenance of Neighbor Cache Entries as described in [RFC6775] section 3.5. The Join Assistant maintains two sets of NCE for each physical interface that it has: one set is for secured neighbors, and the other is for new pledges that wish to join. The storage allocated for pledges will generally be a small fraction of available space. The Join Assistant will garbage collect the different caches according to different thresholds. It MAY chose to free space from the insecure cache to make space for additional secure entries, but it MUST NOT do the opposite.

It sends Enhanced Beacons which are authenticated with the network-wide key ("K2"), but it does not encrypt the Beacons.

In addition, it listen for packets "encrypted" with the well-known "K1" key, and when it receives them, it considers them to be as "Insecure Packets". It MAY also accept unencrypted, unauthenticated packets as being "Insecure Packets"

Non-Join Assistant 6LRs would never accept K1 packets, nor unauthenticated packets. Normal 6LRs and hosts MUST accept unencrypted Enhanced Beacons which can be authenticated with the "K2" key.

In addition to seperating the secured and insecured packets for inbound processing, the Join Assistant also allocates a unique IID for the insecured interface. This IID is used to configure the Link Local address on that virtual interface. This Link Local address is called the Insecure Join Assistant Link Local, or IJALL.

In addition, this IID is combined with the global prefix(es) (as found in various PIO(s) from the routing protocols). This additional address is configured as an alias on the loopback interface such that the Join Assistant can receive packets to this address via secured network. This activity SHOULD generate a routing protocol update (such as an updated DAO). This IID SHOULD be generated using a stable privacy address mechanism as described in [RFC7217]. The easiest is to assign the insecure virtual interface a unique if_index. This new address is called the Pledge Tunnel End Point Address, or PTEPA.

2.4.1. Processing of Insecure Packets

Only the following insecure packets are to be accepted by the Join Assistant:

1. Unicast Neighbor Solicitations.

2. Unicast Link-Local UDP packets with a destination port that map to the Join Assistant’s IPIP proxy.

2.4.1.1. Processing of Insecure Neighbor Solicitation packets

Upon receipt of an insecure unicast NS with an ARO option, the Join Assistant looks up an NCE by the IID contained in the ARO in the insecure cache. If it finds an existing there are three possibilities:

1. a lookup for this entry has previously been completed, and has resulted in a negative result. In this case, a negative ND_NS_JOIN_DECLINED NA is returned. The Join Assistant SHOULD rate limit the number of these messages that it is willing to return.

2. a lookup for this entry has previously been done, and has resulted in a positive result. The NCE entry should be "refresh", to keep it in the cache for a longer period of time, and a new NA returned with a positive status.

3. a lookup for this entry has previously been started, but no result has been received. In this case the Join Assistant SHOULD remain silent. The Join Assistant may wish to send a GRASP M_NOOP message to verify that the connection is still useable if it has not receive any traffic in some time.

If it does not find an existing entry, and there is space for another entry, (or it can make space via garbage collection), then a new entry is created, marked to be "in progress", and a new GRASP 6JOIN query is initiated, see section (#6joingrasp).

3. Join Assistant to Registrar protocol

There are three aspects to the protocols that the Join Assistant uses to communicate about its needs. They are:

1. Discovery of Registrar

2. Notification of new pledge to Registrar

3. Passing of traffic from Pledge to Registar

3.1. Discovery of Registrar

The address of the registrar MAY be determined by other protocols, such as DHCP, RA or RPL extensions, or provisioned into the Join Assistant via other configuration protocol such as 6p.

In order to support fully autonomic operations, the Join Assistant MAY use a GRASP discovery ([I-D.ietf-anima-grasp]) to find the address of the Registrar. [I-D.richardson-anima-6join-discovery] describes the details of the process.

In 6tisch networks multicast is not always available, requiring additional protocol [RFC7731] effort. Instead of doing a multicast GRASP discovery, the Join Assistant SHOULD instead to a TCP connect to the GRASP_LISTEN_PORT on the IP address of the DODAG root (when

RPL is used as the routing protocol for 6tisch), or the ABRO address when another protocol is used. The Join Assistant should then issue the appropriate M_DISCOVERY method using the 6JOIN objective. The GRASP discovery will then reply using the same TCP connection as per Unicast Discovery in [I-D.ietf-anima-grasp] section TBD.

3.2. Notification of a new pledge to Registar

As illustrated in (#joindiagram), when the Join Assistant receives a Neighbor Solication from a pledge, it must notify the Registar of the pledge, indicating to it how to reach the new pledge. The Registrar will respond with a positive acknowledgement if the Registrar is willing/able to accept the pledge. The Registrar will respond with a negative acknowledgement if the provided pledge identity (the IID in the ARO) is not one that the Registrar recognizes as belonging to this network.

The Registrar runs an ASA which is called the 6JOIN ASA (which can be discovered above). This query/response is done using GRASP with the discovered ASA process.

The query process is described in CDDL as:

```
request-6join-query = [M_REQ_NEG, session-id, "6JOIN", [IID, "6p-ipip"]]
IID = bytes .size 8
```

The response from the Registrar is describe as:

```
response-6join-query = [M_END, session-id, [O_ACCEPT]]
```

or for a negative response:

```
response-6join-query = [M_END, session-id, [O_DECLINE]]
```

for the 6p-ipip, the Registrar will need to know where to send the IPIP packets to. The Join Assistant will initiate the TCP connection to the Registrar’s ASA using the IPv6 address associated with the insecure interface on which the pledge is located, i.e. using the PTEPA.

3.3. Passing of traffic from Pledge to Registar

When the Registrar is ready to initiate the pledge into the domain, the Registrar will reach out to the pledge using a secure CoAP protocol (6p). The security is provided using DTLS or EDHOC. As the pledge has only a link-local address, and the Registrar is not co-located on the same layer-2 as the pledge, the traffic must be relayed through the Join Assistant.

To do this the Registrar needs to configure a Link Local address on a virtual inteface which is the same as the PTEPA derived address.

The Registrar then sends it’s traffic (UDP packets with CoAPS inside), inside of an IPIP header to the Join Assistant. The outer IP header is from the Registrar to the PTEPA. The inner IP is from the link local address configured above, and the destination is the Link Local address of the pledge.

The Registrar knows the IJALL by taking the IID from the connection address above, and knows the Link Local of the pledge from the IID in the objective message.

The Join Assistant, upon receipt of the IPIP traffic from the Registar on it’s PTEPA, then decapsulates it and forwards it on the appropriate. (This is identical code to decapsulation of IPIP headers as specified in [I-D.ietf-roll-useofrplinfo].

The Join Assistant, upon receiving traffic from the pledge to the IJALL, it encapsulates it into an IPIP header, setting the source of this outer header to the PTEPA, and the destination being the Registrar.

The Join Assistant can do this for as many pledges as the Registrar decides to communicate with, without using any additional per-pledge state other than the obligatory Neighbor Cache Entries needed to translate L3 addresses to L2 addresses.

4. 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.

4.1. Privacy Considerations for Production network

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

4.2. 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 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 successful 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.

4.2.1. 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.

4.3. Privacy Considerations for Join Assistant

5. Security Considerations

6. 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)

7. Protocol Definition

8. References

8.1. 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 over Networks of Resource-Constrained Nodes", draft-ietf-6lo-privacy-considerations-03 (work in progress), September 2016.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-16 (work in progress), June 2016.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-03 (work in progress), June 2016.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-07 (work in progress), September 2016.

[I-D.ietf-netconf-zerotouch]
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-09 (work in progress), July 2016.

[I-D.richardson-6lo-ra-in-ie]
Richardson, M., "802.15.4 Informational Element encapsulation of ICMPv6 Router Advertisements", draft-richardson-6lo-ra-in-ie-00 (work in progress), October 2016.

[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>.

[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>.

8.2. 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-08 (work in progress), September 2016.

[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>.

Appendix A. appendix

insert appendix here

Author’s Address

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2017

M. Vucinic, Ed.
Inria
J. Simon
Linear Technology
K. Pister
University of California Berkeley
October 25, 2016

Minimal Security Framework for 6TiSCH
draft-vucinic-6tisch-minimal-security-00

Abstract

This draft describes the minimal mechanisms required to support secure initial configuration in a device being added to a 6TiSCH network. The goal of this configuration is to set link-layer keys, and to establish a secure session between each joining node and the JCE who may use that to further configure the joining device. Additional security behaviors and mechanisms may be added on top of this minimal framework.

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

Table of Contents

1. Introduction 3
2. Terminology 3
3. Join Overview 4
3.1. Step 1 - Enhanced Beacon 5
3.2. Step 2 - Neighbor Discovery 5
3.3. Step 3 - Security Handshake 5
3.3.1. Pre-Shared Key 5
3.3.2. Asymmetric Keys 5
3.4. Step 4 - Join Request 6
3.5. Step 5 - Join Response 6
4. Protocol Specification 6
4.1. Proxy Operation of JA 7
4.1.1. Implementation of origin_info 7
4.2. OSCOAP Security Context Instantiation 7
4.3. Implementation of Join Request 8
4.4. Implementation of Join Response 8
5. Link-layer requirements 9
5.1. Well-known beacon authentication key 10
5.2. Private beacon authentication key 10
6. Asymmetric Keys 10
7. Security Considerations 10
8. Privacy Considerations 11
9. IANA Considerations 12
10. Acknowledgments 12
11. References 12
11.1. Normative References 12
11.2. Informative References 12
11.3. External Informative References 14
Appendix A. Example 14
Authors' Addresses 16

1. Introduction

When a previously unknown device seeks admission to a 6TiSCH [RFC7554] network (to "join"), it first needs to synchronize to the network. The device then configures its 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 DTLS [RFC6347] or OSCOAP [I-D.ietf-core-object-security]. Once the application requirements are known, the device 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 describes the mechanisms comprising a minimal feature set for a device to join a 6TiSCH network, up to the point where it can establish a secure session with an Internet host.

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

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

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

2. Terminology

The reader is expected to be familiar with the terms and concepts defined in [I-D.ietf-6tisch-terminology], [RFC7252], and [I-D.ietf-core-object-security]. The entities participating in the protocol that is specified in this document are:

- JN: Joining node - the device attempting to join a particular 6TiSCH network.
- JCE: Join coordinating entity - central entity responsible for authentication and authorization of joining nodes.
- JA: Join assistant - the device within radio range of the JN that generates Enhanced Beacons (EBs) and facilitates end-to-end communications between the JN and JCE.

3. Join Overview

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

1. The JN listens for an Enhanced Beacon (EB) frame [IEEE802154-2015]. 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 Assistant (JA) for the JN, and when it can expect to receive a frame.
2. The JN configures its link-local IPv6 address and advertises it to JA.
3. The JN sends packets to the JA device in order to securely identify itself to the network. These packets are directed to the Join Coordination Entity (JCE), which may be the JA or another device.
4. The JN receives one or more packets from JCE (via the JA) that sets up one or more link-layer keys used to authenticate subsequent transmissions to peers.

From the joining node’s perspective, minimal joining is a local phenomenon - the JN only interacts with the JA, and it need not know how far it is from the DAG root, or how to route to the JCE. 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:

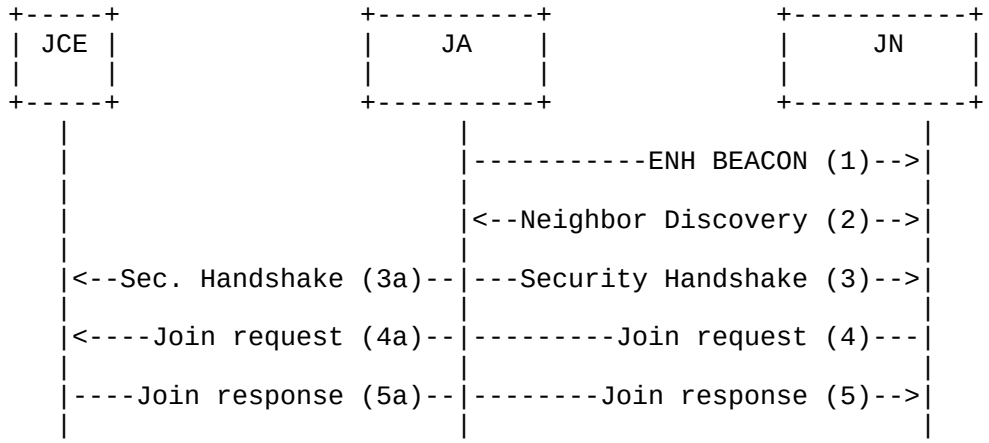


Figure 1: Message sequence for join protocol.

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

3.1. Step 1 - Enhanced Beacon

The JN hears an EB from the JA and synchronizes itself to the joining schedule using the cells contained in the EB. At this point the JN MAY proceed to step 2, or continue to listen for additional EBs. If more than one EB is heard, the JN MAY use a metric based on DAG rank and received signal level of the EB, or other factors to decide which JA to use for the security handshake in step 3. Details on how a JN chooses the JA are out of scope of this specification.

3.2. Step 2 - Neighbor Discovery

At this point, JN forms its link-local IPv6 address based on EUI64 and MAY further follow the Neighbor Discovery (ND) process described in Section 5 of [RFC6775].

3.3. Step 3 - Security Handshake

The security handshake between JN and JCE uses Ephemeral Diffie-Hellman over COSE (EDHOC) [I-D.selander-ace-cose-ecdhe] to establish the shared secret used to encrypt the join request and join response.

The security handshake step is OPTIONAL in case PSKs are used, while it is REQUIRED for RPKs and certificates. In case the handshake step is omitted, the shared secret used for protection of the join request and join response in the next step is the PSK. This means that the protocol trades off perfect forward secrecy for reduced traffic load between JN and JCE. 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 JN, however, would also imply the compromise of the same keying material, as it is likely to be found in node’s memory.

3.3.1. Pre-Shared 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 JCE and JN, but requires pre-configuration of the shared key on both devices.

It is REQUIRED to use unique PSKs for each JN.

3.3.2. 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 JN and JCE need to receive and

validate each other's public key certificate. When RPKs are pre-configured at JN and JCE, they can directly proceed to the handshake.

3.4. Step 4 - Join Request

The join request is sent from the JN to the JA using the slot information from the EB, and forwarded to the JCE.

The join request is authenticated/encrypted end-to-end using AES-CCM-16-64-128 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, JN's EUI64 and a monotonically increasing counter initialized to 0 when first starting.

3.5. Step 5 - Join Response

The join response is sent from the JCE to the JN through JA that serves as a stateless relay. Packet containing the join response travels on the path from JCE to JA using pre-established routes in the network. The JA delivers it to the JN using the slot information from the EB. JA 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 using AES-CCM-16-64-128 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, JN's EUI64 and a monotonically increasing counter matching that of the join request.

The join response contains one or more (per-peer) link-layer key(s) K2 that the JN will use for subsequent communication. It optionally also contains an IEEE 802.15.4 short-address [IEEE802154-2015] assigned to JN by JCE.

4. Protocol Specification

The join protocol in Figure 1 is implemented over Constrained Application Protocol (CoAP) [RFC7252]. JN plays the role of a CoAP client, JCE the role of a CoAP server, while JA implements CoAP forward proxy functionality [RFC7252]. Since JA is 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 JN that is not yet part of the network.

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

4.1. Proxy Operation of JA

JN designates a JA as a proxy by including in the CoAP requests to the JA the Proxy-Scheme option with value "coap" (CoAP-to-CoAP proxy). JN MUST include the Uri-Host option with its value set to the well-known JCE's alias - "6tisch.jce". JN does not need to learn the actual IPv6 address of JCE at any time during the join protocol. JA resolves the address by performing a GET request at "/jce" resource of its parent in the DODAG.

Note that the CoAP proxy by default keeps state information in order to forward the response towards the originator of the request. 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, as in the case of JA, this makes it prone to Denial of Service (DoS) attacks, due to the limited memory.

In order to facilitate a stateless implementation of JA proxying, JN shall encode in the CoAP message the information necessary for the JA to send the response back - "origin_info". For this purpose, JN uses the "Context Identifier (Cid)" parameter of OSCOAP's security context structure. Context Identifier is sent in clear, readable by JA, and MUST be echoed back in the response from JCE. This makes it possible to implement JA's CoAP proxy in a stateless manner. It also allows JCE to look up the right security context for communication with a given JN.

4.1.1. Implementation of origin_info

The origin_info is implemented as a CBOR [RFC7049] array object containing:

- EUI64: JN's EUI64 address
- source_port: JN's UDP source port
- token: JN's CoAP token

```
origin_info = [
    EUI64 : bstr,
    source_port : uint,
    token : uint
]
```

4.2. OSCOAP Security Context Instantiation

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

- Context Identifier (Cid) MUST be the origin_info object wrapped as a byte string (bstr).
- Algorithm MUST be set to AES-CCM-16-64-128 from [I-D.ietf-cose-msg]. CoAP messages are therefore protected with an 8-byte CCM authentication tag and the algorithm uses 13-byte long nonces.
- Base key (base_key) MUST be the secret generated by the run of EDHOC, or the PSK in case EDHOC step was omitted.
- Sender ID of JN MUST be set to 0x00, while the ID of JCE MUST be set to 0x01.

The hash algorithm that instantiates HKDF MUST be SHA-256 [RFC4231]. 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 JCEs 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 JN tries to join one network at a time.

4.3. Implementation of Join Request

Join Request message SHALL be mapped to a CoAP request:

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

4.4. Implementation of Join Response

If OSCOAP processing is a success, Join Response message SHALL be a CoAP response:

- The response Code is 2.05 (Content).
- The payload is a CBOR array containing, in order:
  - COSE Key Set [I-D.ietf-cose-msg]. Each key in the Key Set SHALL be a symmetric key. A key that is present in the Key Set and does not have an identifier is assumed to be "K2" link-layer key from [I-D.ietf-6tisch-minimal]. Parameter "kid" of the COSE Key structure SHALL be used to denote pair-wise keys

if present, where the value SHALL be set to the address of the corresponding peer.

* Optional byte string representing IEEE 802.15.4 short address assigned to JN. If the length of the byte string is different than 2 bytes, the implementation SHOULD ignore it.

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

In case JCE determines that JN 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, JN SHALL attempt to join the next advertised 6TiSCH network. If all join attempts have failed at JN, JN SHOULD signal to the user by an out-of-band mechanism the presence of an error condition.

5. Link-layer requirements

All frames in a 6TiSCH network MUST use link-layer frame security. The frame security options MUST include frame authentication, and MAY include frame encryption.

In order for the JN to be able to validate that the Enhanced Beacon frame is coming from a 6TiSCH network, EB frames are authenticated at the link layer using CCM* per [IEEE802154-2015]. 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 JA for EBs) address, as shown in Figure 2:

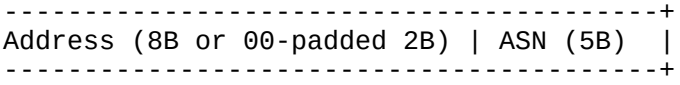


Figure 2: Link-layer CCM* nonce construction

The JN uses the initial key K1 [I-D.ietf-6tisch-minimal] until it is configured with a new link-layer key K2 as described above. JA SHOULD secure/verify DATA and ACKNOWLEDGMENT frames destined/originated at JN with K1 only during the duration of the join process. How JA learns whether the join process is ongoing is out of scope of this specification.

As the EB itself does not contain security information, where the link key is known, an attacker may craft a frame that appears to be a valid EB, since the JN can neither know the ASN a priori nor verify

the address of the JA. This permits a Denial of Service (DoS) attack at the JN. Beacon authentication keys are discussed in Section 5.1 and Section 5.2.

5.1. Well-known beacon authentication key

For zero-touch operation, where any 6TiSCH device can attempt to join any 6TiSCH network out of the box, a well-known EB link-layer key MUST be used. The value of this key is specified in [I-D.ietf-6tisch-minimal].

5.2. Private beacon authentication key

Some pre-configuration MAY be done when the device is manufactured or designated for a specific network (i.e. the network is one-touch) or a network operator may not wish to allow arbitrary devices to try to join. A private (per-vendor, or per-installation) EB link-layer key MAY be used in place of a well-known key to create a private network.

6. Asymmetric Keys

Certificates or pre-configured RPKs may be used to exchange public keys between the JN and JCE. The key pair is generated using elliptic curve secp256r1, and the certificate containing the public key is signed using ECDSA. 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 JCE or JN as appropriate.

7. Security Considerations

In case PSKs are used, this document mandates that JN and JCE are pre-configured with unique keys. The uniqueness of generated nonces is guaranteed under the assumption of unique EUI64 identifiers for each JN. Note that the address of the JCE does not take part in nonce construction. Therefore, even under the assumption of a PSK shared by a group of nodes, the nonces constructed as part of the different responses are unique. The design differentiates between nonces constructed for requests and nonces constructed for responses by different sender identifiers (0x00 for JN and 0x01 for JCE).

Being a stateless relay, JA blindly forwards the join traffic into the network. While the exchange between JN and JA takes place over a shared cell, join traffic is forwarded using dedicated cells on the JA to JCE path. In case of distributed scheduling, the join traffic

may therefore cause intermediate nodes to request additional bandwidth. Because the relay operation of JA is implemented at the application layer, JA is the only hop on the JA-6LBR path that can distinguish join traffic from regular IP traffic in the network. It is therefore permitted to implement rate limiting at JA.

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

Because the well-known beacon authentication key does not provide any security, it is feasible for an attacker to generate EBs that will get accepted at JN. At the time of the join, JN has no means of verifying the content in the EB and has to accept it at "face value". As the join response message in such cases will either fail the security check or time out, JN 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 JN and so the time spent in "minimal" [I-D.ietf-6tisch-minimal] duty cycle mode. The permitted practice is to use a private, per-installation beacon authentication key.

8. Privacy Considerations

This specification relies on the uniqueness of EUI64 that is transferred in clear as part of the security context identifier. 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 JCE to JN. 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].

9. IANA Considerations

There is no IANA action required for this document.

10. 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. The authors would also like to thank Francesca Palombini and Ludwig Seitz for participating in the discussions that have helped shape the document.

11. References

11.1. 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>.

[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>.

[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>.

[I-D.ietf-cose-msg] Schaad, J., "CBOR Object Signing and Encryption (COSE)", draft-ietf-cose-msg-20 (work in progress), October 2016.

[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-00 (work in progress), October 2016.

11.2. 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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[I-D.ietf-6tisch-minimal] Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-16 (work in progress), June 2016.

[I-D.ietf-6tisch-6top-protocol] Wang, Q. and X. Vilajosana, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-02 (work in progress), July 2016.

[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.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-02 (work in progress), July 2016.

11.3. External Informative 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.

Appendix A. Example

Figure 3 illustrates a join protocol exchange in case PSKs are used. JN 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 JCE in the Uri-Host option. The example assumes a JA that is already aware of JCE's IPv6 address and does not need to resolve the well-known "6tisch.jce" host name. Triggered by the presence of Proxy-Scheme option, JA forwards the request to the JCE. Once JCE receives the request, it looks up the correct context based on the context identifier (cid) field. It reconstructs OSCOAP's external Additional Authenticated Data (AAD) needed for verification based on:

- Version field of the received CoAP header.
- Code field of the received CoAP header.
- Algorithm being the AES-CCM-16-64-128 from [I-D.ietf-cose-msg].
- Request URI reconstructed following [I-D.ietf-core-object-security].

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 JN. Once JA receives the response, it looks up and decodes the cid field in order to decide where to forward it. JA constructs the CoAP response to JN by setting the CoAP token to the value decoded from cid and constructs the link-local IPv6 address of JN from the EUI64 address found in the cid. Note that JA does not posses 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 JN using OSCOAP processing rules. Namely, to verify the response JN reconstructs the AAD based on:

- Version field of the received CoAP header.
- Code field of the received CoAP header.

- Algorithm being the AES-CCM-16-64-128 from [I-D.ietf-cose-msg].
- Transaction identifier (Tid) of the corresponding CoAP request. Tid contains the context identifier (origin_info object), Sender ID (0x00 for JN), and Sender Sequence number (set to 1 in the example).

In addition to AAD, JN also uses the explicit, protected fields in the COSE message, present in the payload of the response. For more details, see [I-D.ietf-core-object-security] and [I-D.ietf-cose-msg].

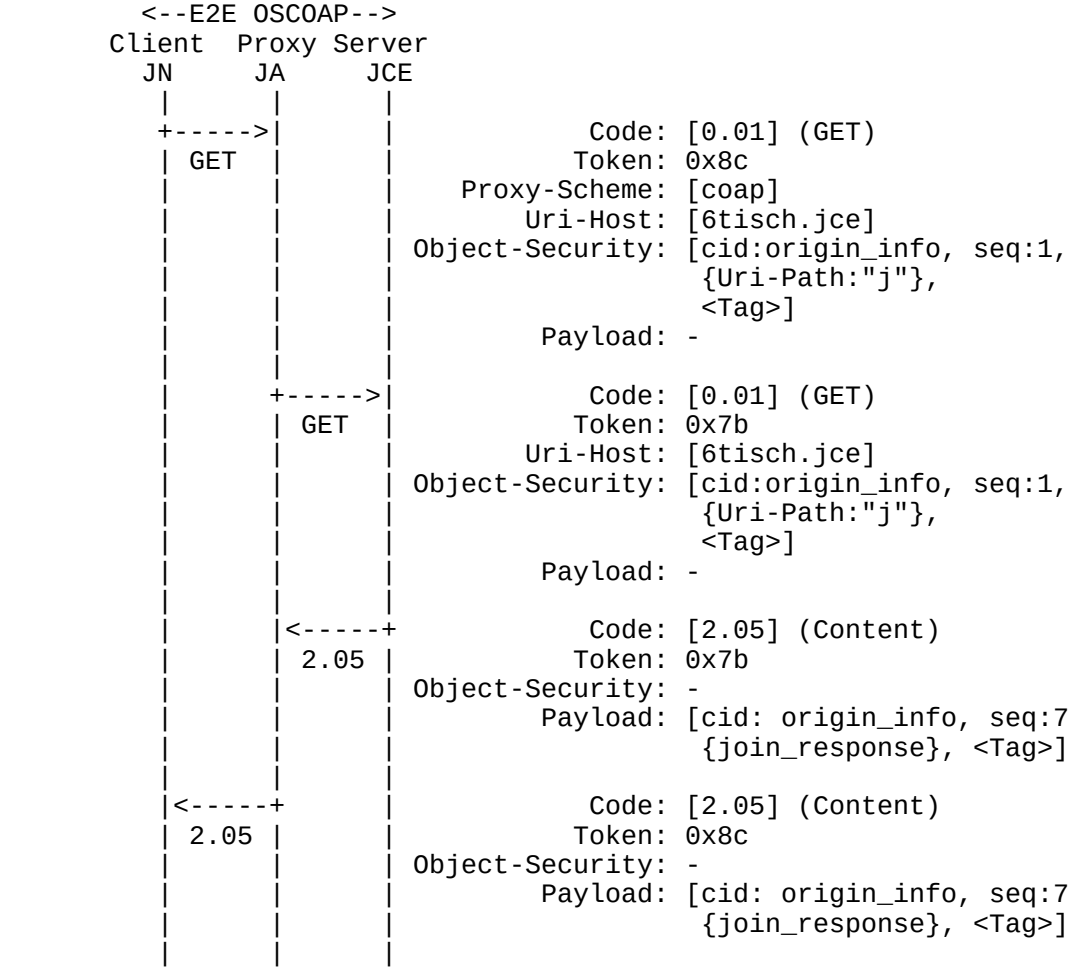


Figure 3: Example of a join protocol exchange with a PSK. {} denotes encryption and authentication, [] denotes authentication.

Where origin_info and join_response are as follows.

```
origin_info:
[
     h'00170d00060d9f0e', / JN's EUI64 /
     49152, / JN's UDP source port /
     0x8c   / JN's CoAP token /
]
```

Encodes to h'834800170d00060d9f0e19c000188c' with a size of 15 bytes.

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

Encodes to h'8281a201042050e6bf4287c2d7618d6a9687445ffd33e642af93' with a size of 26 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
490 Cory Hall
Berkeley, California 94720
USA

Email: kpister@eecs.berkeley.edu