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

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.

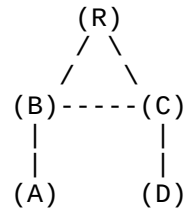

Figure 1: A simple 6TiSCH network.

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.

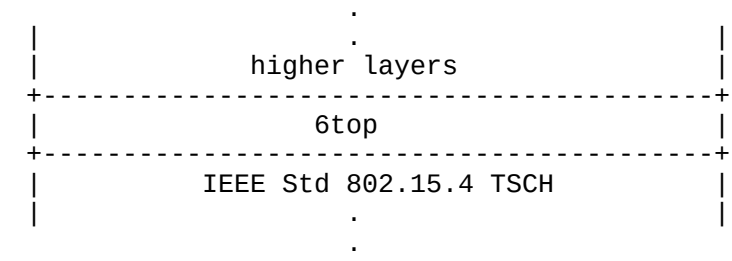

Figure 2: The 6top sublayer in the protocol stack.

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.

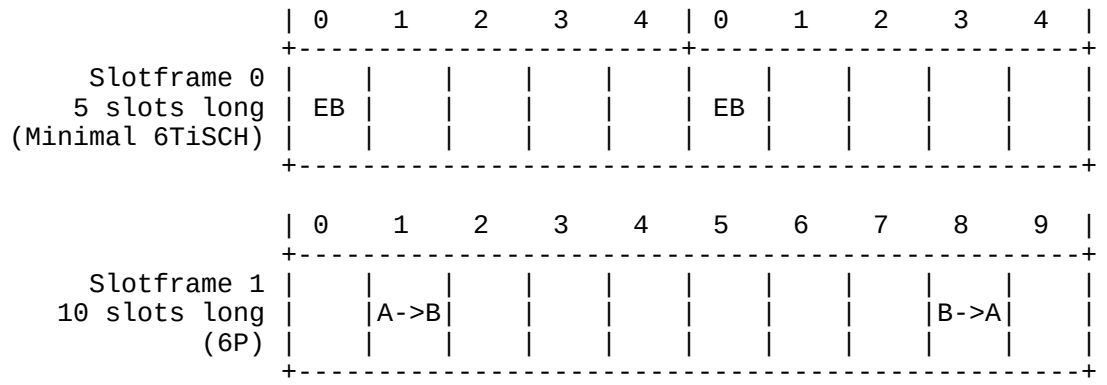

Figure 3: 2-slotframe structure when using 6P alongside the Minimal 6TiSCH Configuration.

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.

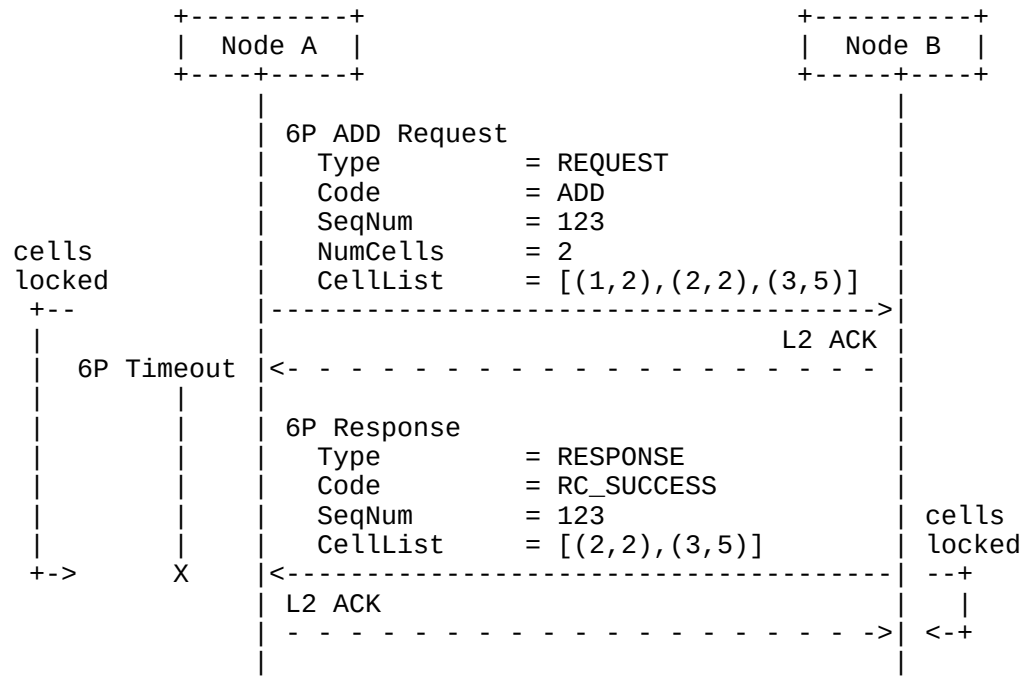

Figure 4: An example 2-step 6P Transaction.

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.

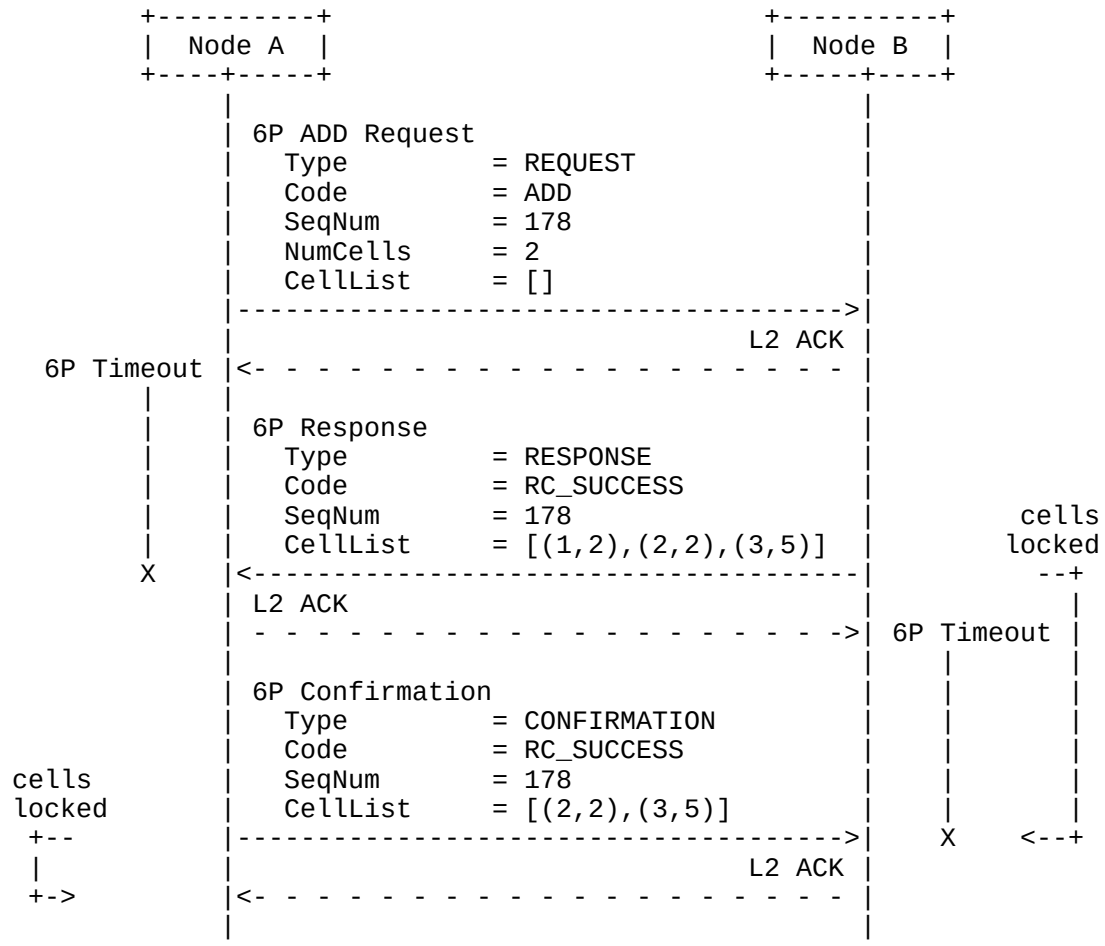

Figure 5: An example 3-step 6P Transaction.

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.

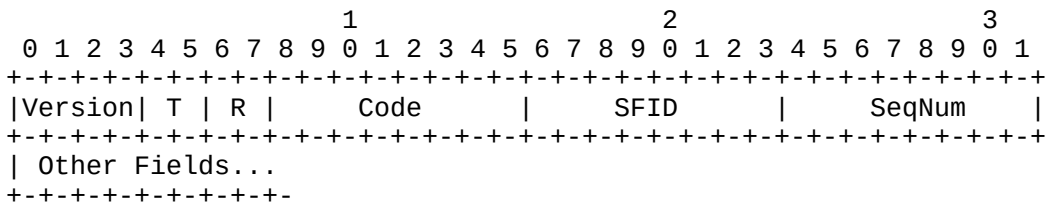

Figure 6: Generic 6P Message Format.

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:

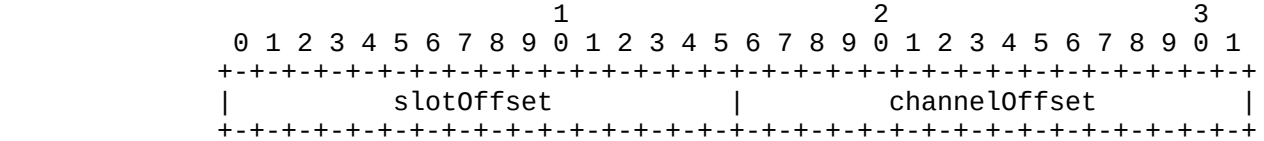

Figure 9: 6P Cell Format.

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.

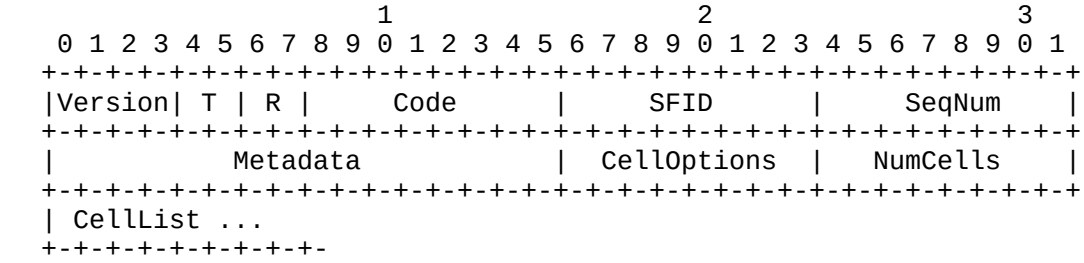

Figure 10: 6P ADD Request Format.

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.

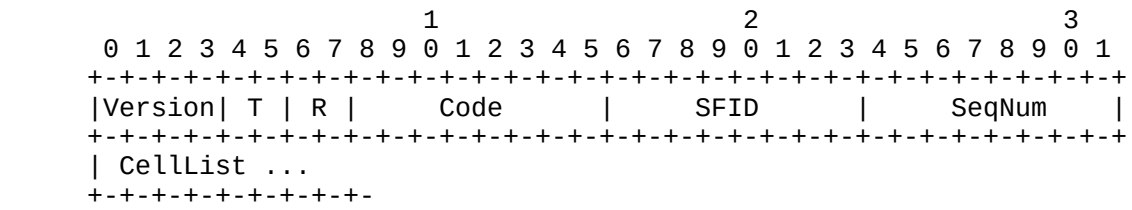

Figure 11: 6P ADD Response and Confirmation Formats.

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.

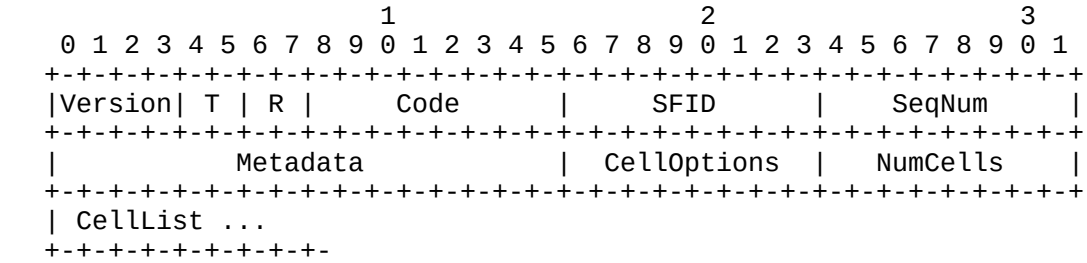

Figure 12: 6P DELETE Request Format.

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.

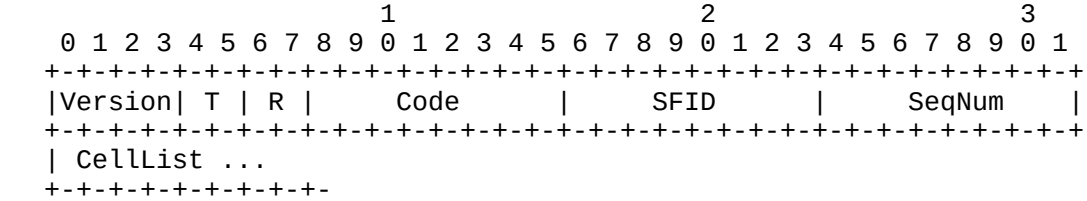

Figure 13: 6P DELETE Response and Confirmation Formats.

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.

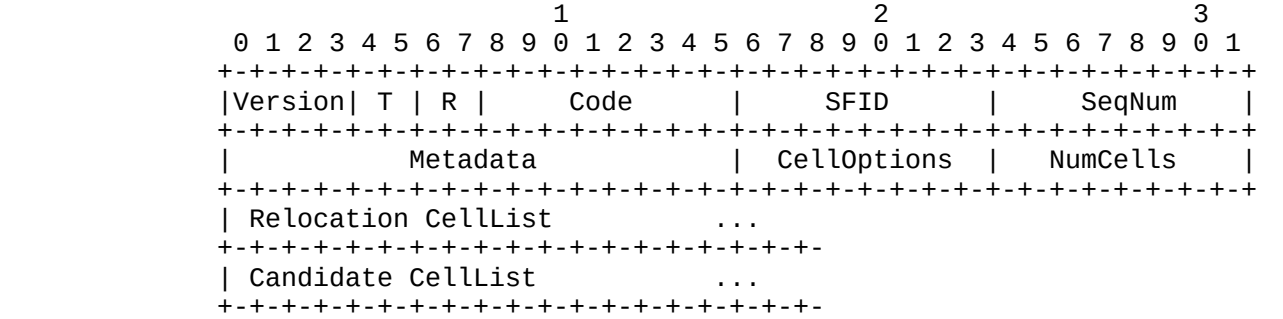

Figure 14: 6P RELOCATE Request Format.

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.

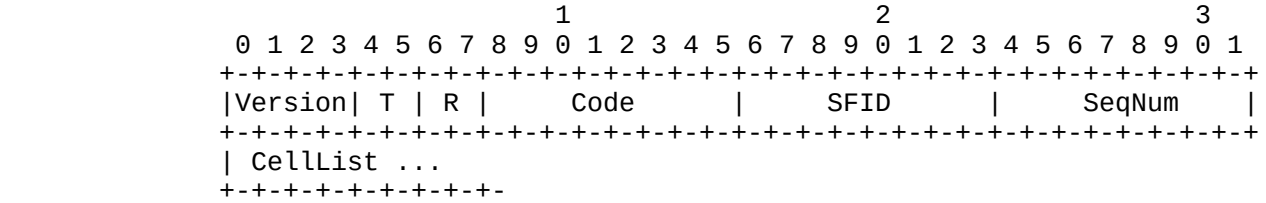

Figure 15: 6P RELOCATE Response and Confirmation Formats.

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.

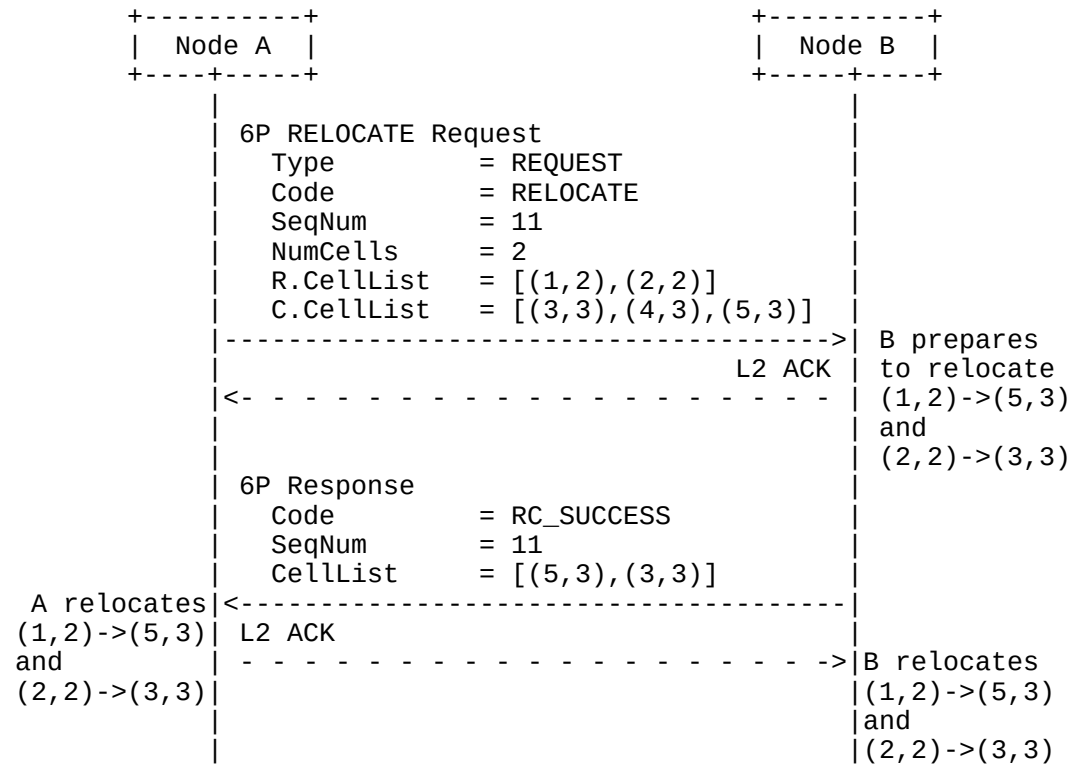

Figure 16: Example of a successful 2-step 6P RELOCATION Transaction.

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

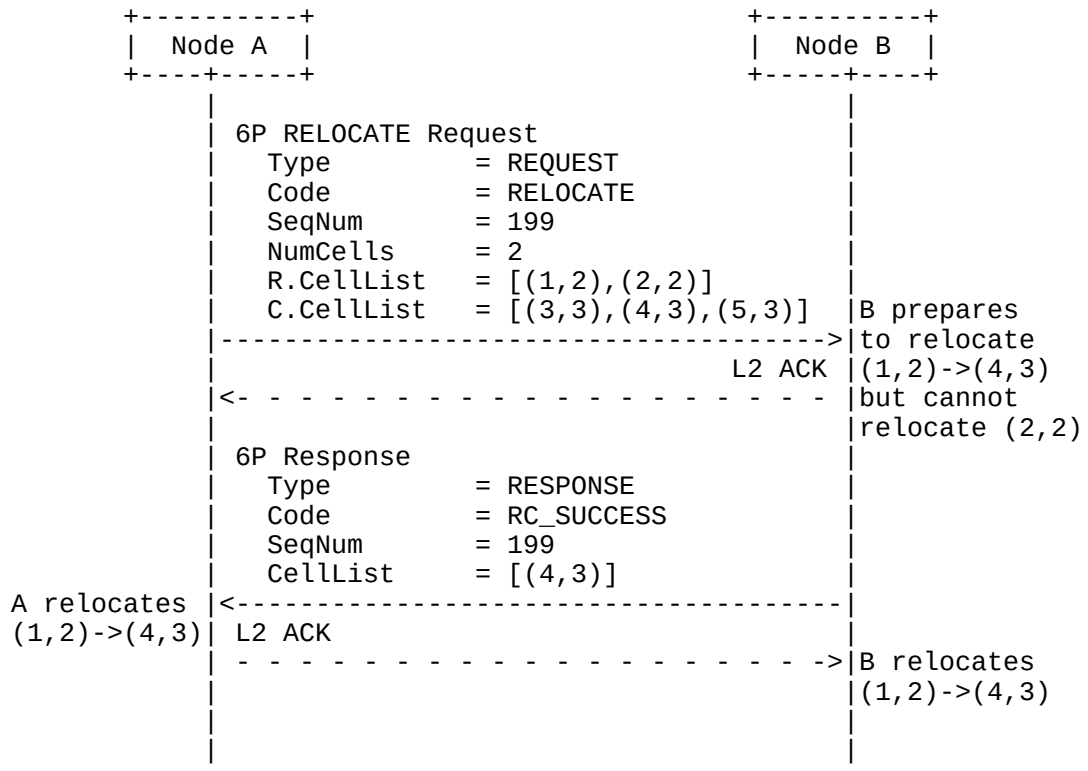

Figure 17: Example of a partially successful 2-step 6P RELOCATION Transaction.

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

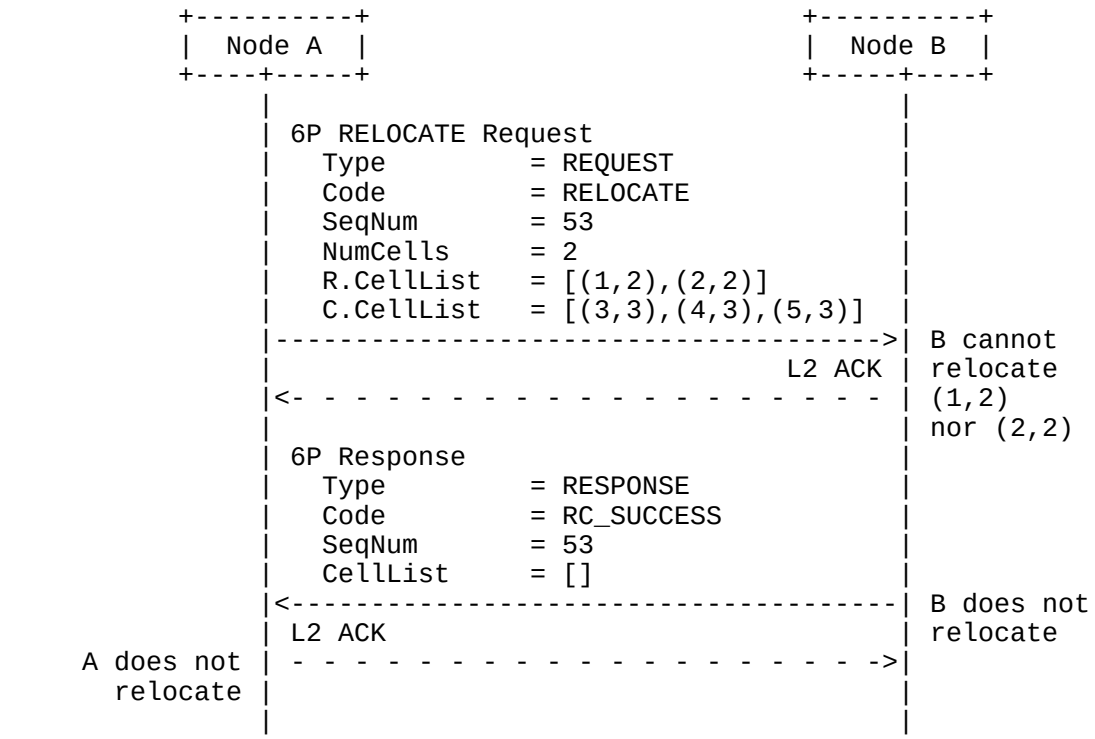

Figure 18: Failed 2-step 6P RELOCATION Transaction Example.

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

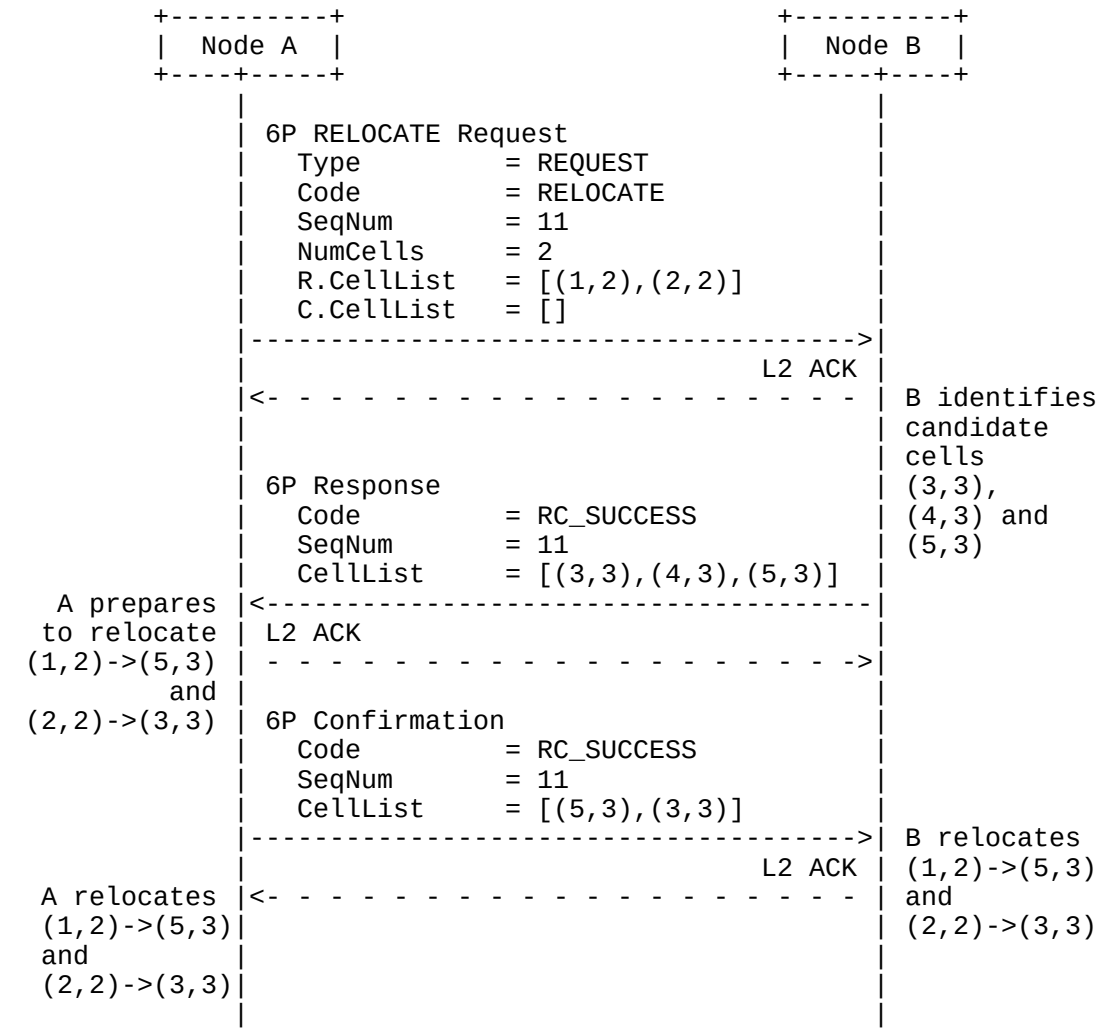

Figure 19: Example of a successful 3-step 6P RELOCATION Transaction.

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.

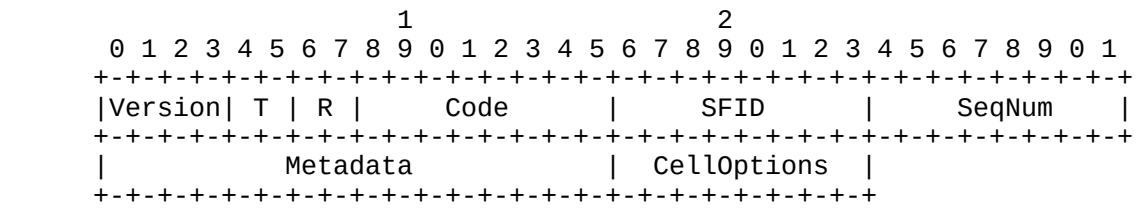

Figure 20: 6P COUNT Request Format.

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.

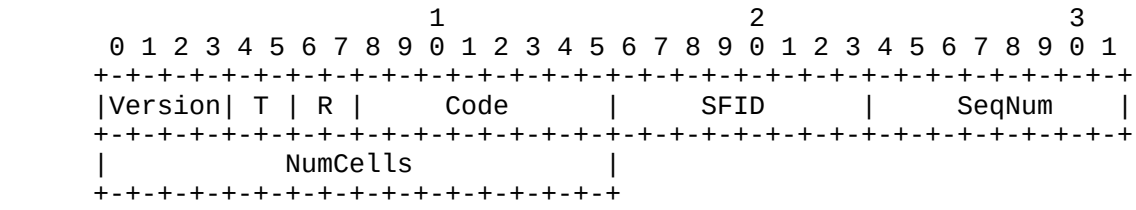

Figure 21: 6P COUNT Response Format.

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.

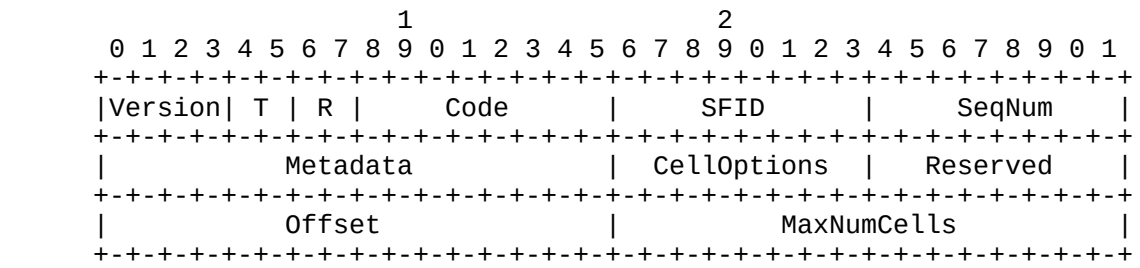

Figure 22: 6P LIST Request Format.

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.

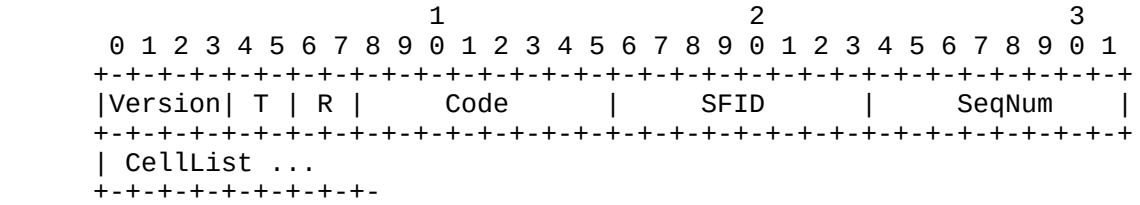

Figure 23: 6P LIST Response Format.

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.

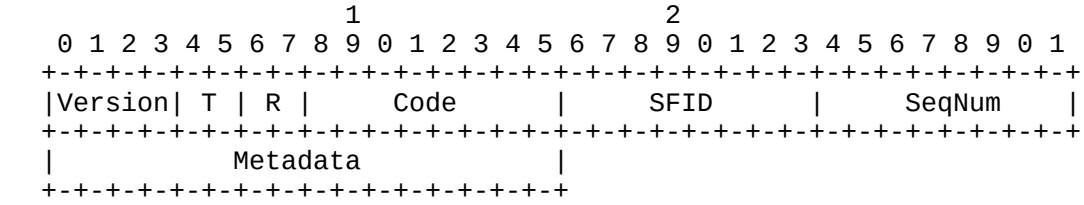

Figure 24: 6P CLEAR Request Format.

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.

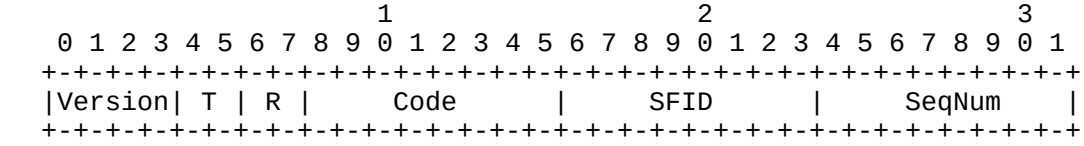

Figure 25: 6P CLEAR Response Format.

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.

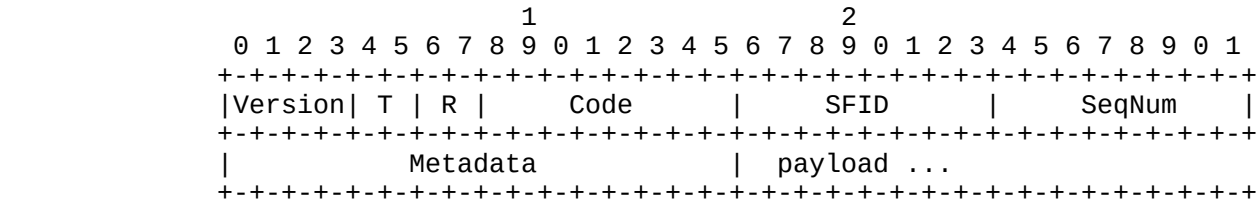

Figure 26: 6P SIGNAL Request Format.

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.

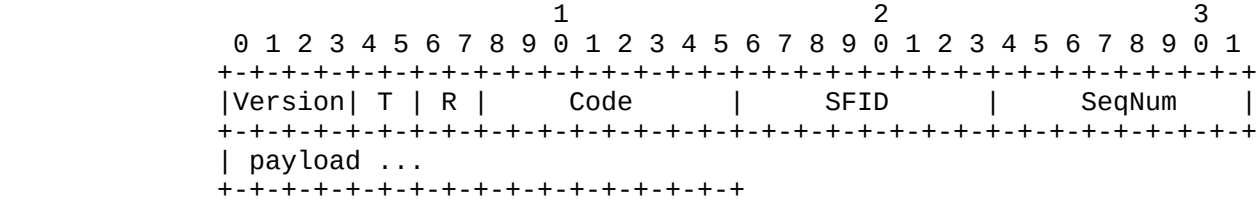

Figure 27: 6P SIGNAL Response Format.

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.

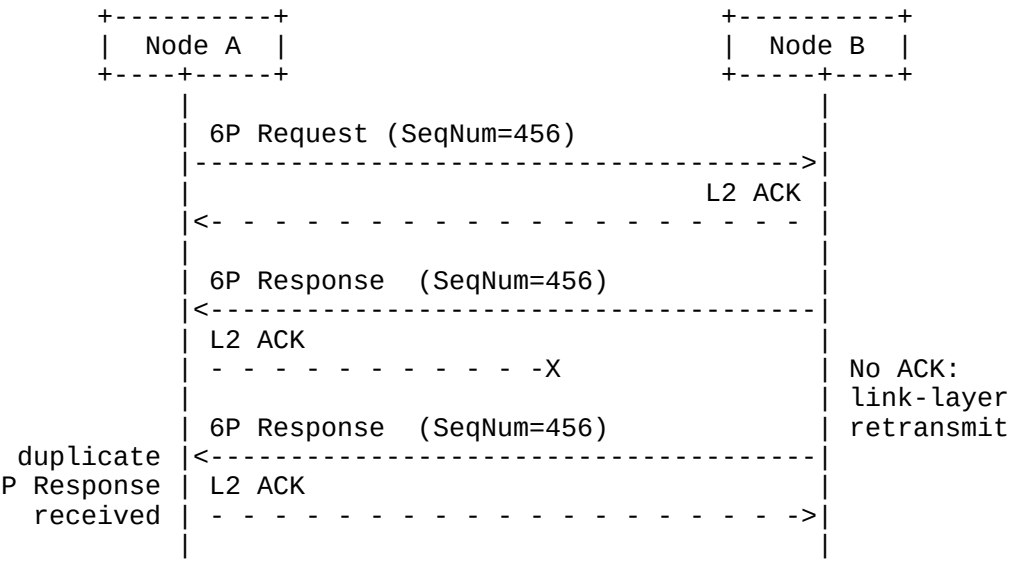

Figure 29: Example duplicate 6P message.

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.

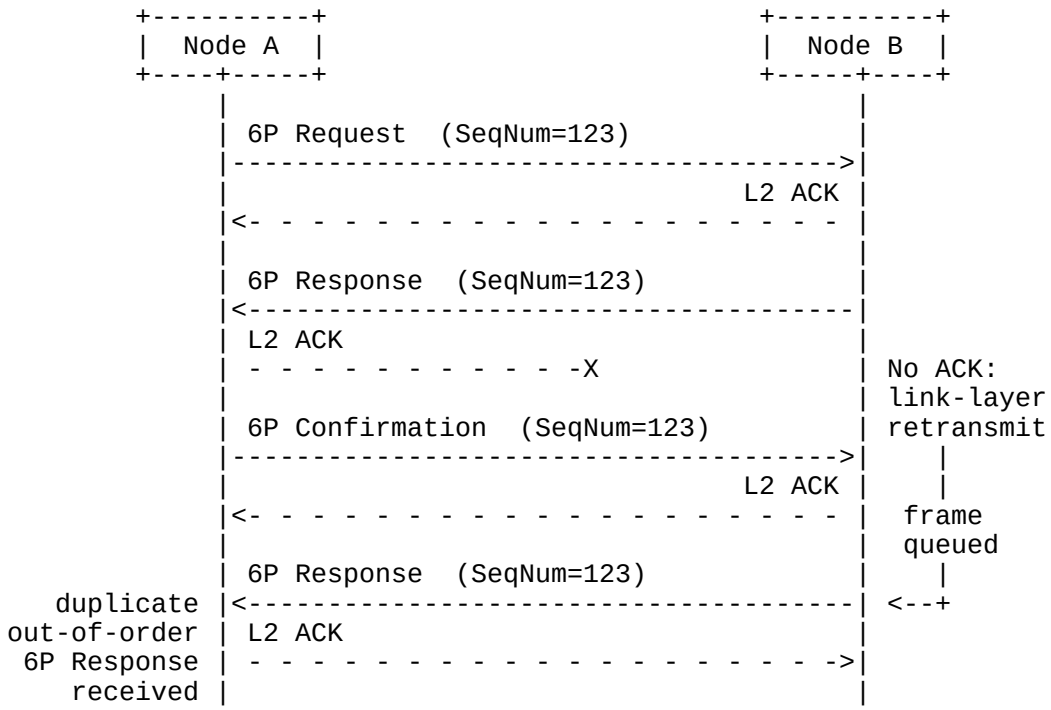

Figure 30: Example out-of-order duplicate 6P message.

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.

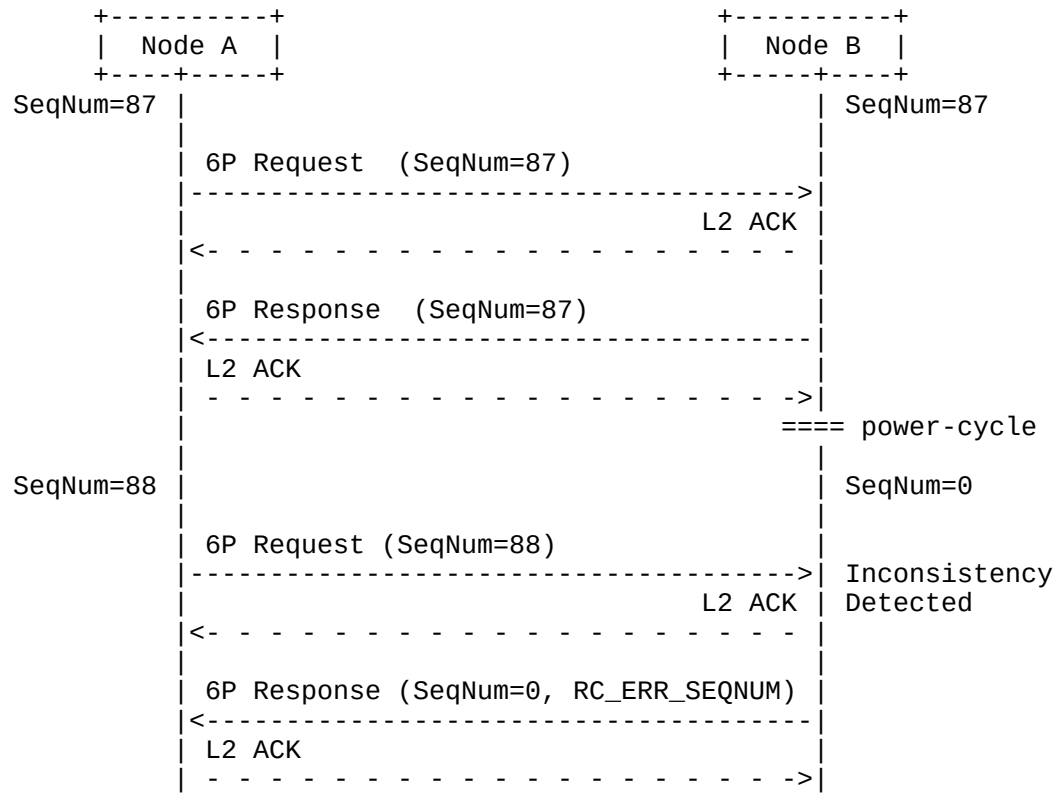

Figure 31: Example of inconsistency because of node B reset. Detected by node B

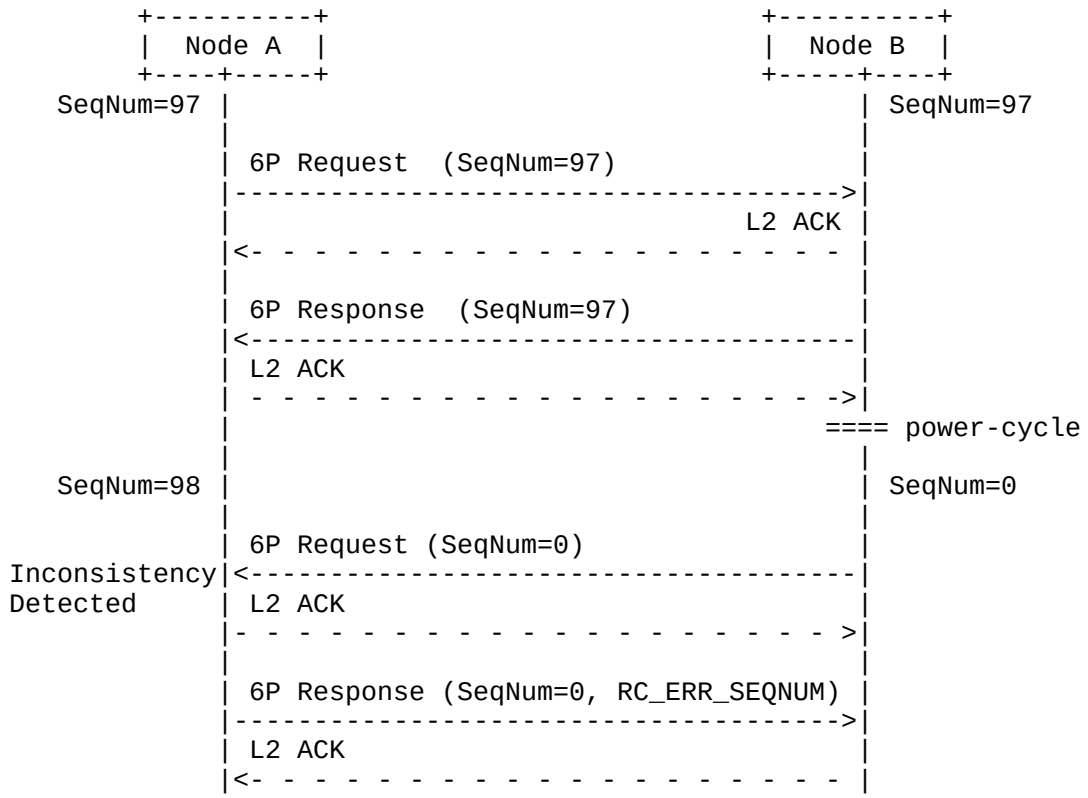

Figure 32: Example of inconsistency because node B resets. Detected by node A

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.

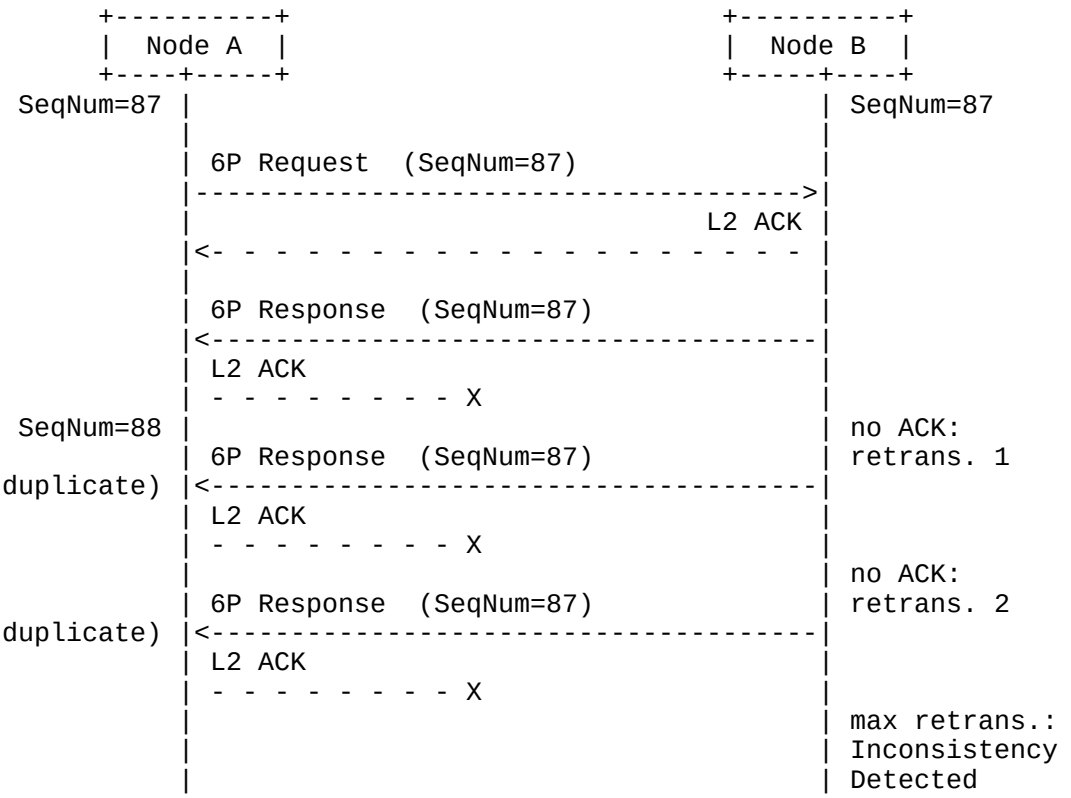

Figure 33: Example inconsistency because of maximum link-layer retransmissions (here 2).

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: Informational
Expires: 30 May 2021

P. Thubert, Ed.
Cisco Systems

26 November 2020

An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4
draft-ietf-6tisch-architecture-30

Abstract

This document describes a network architecture that provides low-latency, low-jitter and high-reliability packet delivery. It combines a high-speed powered backbone and subnetworks using IEEE 802.15.4 time-slotted channel hopping (TSCH) to meet the requirements of LowPower wireless deterministic applications.

Table of Contents

1. Introduction

Wireless Networks enable a wide variety of devices of any size to get interconnected, often at a very low marginal cost per device, at any range, and in circumstances where wiring may be impractical, for instance on fast-moving or rotating devices.

On the other hand, Deterministic Networking maximizes the packet delivery ratio within a bounded latency so as to enable mission-critical machine-to-machine (M2M) operations. Applications that need such networks are presented in [RFC8578]. The considered applications include Professional Media, Industrial Automation Control Systems (IACS), building automation, in-vehicle command and control, commercial automation and asset tracking with mobile scenarios, as well as gaming, drones and edge robotic control, and home automation applications.

The Timeslotted Channel Hopping (TSCH) [RFC7554] mode of the IEEE Std. 802.15.4 [IEEE802154] Medium Access Control (MAC) was introduced with the IEEE Std. 802.15.4e [IEEE802154e] amendment and is now retrofitted in the main standard. For all practical purposes, this document is expected to be insensitive to the revisions of that standard, which is thus referenced without a date. TSCH is both a Time-Division Multiplexing and a Frequency-Division Multiplexing technique whereby a different channel can be used for each transmission, and that allows to schedule transmissions for deterministic operations, and applies to the slower and most energy constrained wireless use cases.

The scheduled operation provides for a more reliable experience which can be used to monitor and manage resources, e.g., energy and water, in a more efficient fashion.

Proven Deterministic Networking standards for use in Process Control, including ISA100.11a [ISA100.11a] and WirelessHART [WirelessHART], have demonstrated the capabilities of the IEEE Std. 802.15.4 TSCH MAC for high reliability against interference, low-power consumption on well-known flows, and its applicability for Traffic Engineering (TE) from a central controller.

To enable the convergence of Information Technology (IT) and Operational Technology (OT) in Low-Power Lossy Networks (LLNs), the 6TiSCH Architecture supports an IETF suite of protocols over the IEEE Std. 802.15.4 TSCH MAC to provide IP connectivity for energy and otherwise constrained wireless devices.

The 6TiSCH Architecture relies on IPv6 [RFC8200] and the use of routing to provide large scaling capabilities. The addition of a high-speed federating backbone adds yet another degree of scalability to the design. The backbone is typically a Layer-2 transit Link such as an Ethernet bridged network, but it can also be a more complex routed structure.

The 6TiSCH Architecture introduces an IPv6 Multi-Link subnet model that is composed of a federating backbone and a number of IEEE Std. 802.15.4 TSCH low-power wireless networks federated and synchronized by Backbone Routers. If the backbone is a Layer-2 transit Link then the Backbone Routers can operate as an IPv6 Neighbor Discovery (IPv6 ND) [RFC4861] proxy.

The 6TiSCH Architecture leverages 6LoWPAN [RFC4944] to adapt IPv6 to the constrained media and RPL [RFC6550] for the distributed routing operations.

Centralized routing refers to a model where routes are computed and resources are allocated from a central controller. This is particularly helpful to schedule deterministic multihop transmissions. In contrast, Distributed Routing refers to a model that relies on concurrent peer to peer protocol exchanges for TSCH resource allocation and routing operations.

The architecture defines mechanisms to establish and maintain routing and scheduling in a centralized, distributed, or mixed fashion, for use in multiple OT environments. It is applicable in particular to highly scalable solutions such as used in Advanced Metering Infrastructure [AMI] solutions that leverage distributed routing to enable multipath forwarding over large LLN meshes.

2. Terminology

2.1. New Terms

The draft does not reuse terms from the IEEE Std. 802.15.4 [IEEE802154] standard such as "path" or "link" which bear a meaning that is quite different from classical IETF parlance.

This document adds the following terms:

6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4): 6TiSCH defines an adaptation sublayer for IPv6 over TSCH called 6top, a set of protocols for setting up a TSCH schedule in distributed approach, and a security solution. 6TiSCH may be extended in the future for other MAC/PHY pairs providing a service similar to TSCH.

6top (6TiSCH Operation Sublayer): The next higher layer of the IEEE Std. 802.15.4 TSCH MAC layer. 6top provides the abstraction of an IP link over a TSCH MAC, schedules packets over TSCH cells, and exposes a management interface to schedule TSCH cells.

6P (6top Protocol): The protocol defined in [RFC8480]. 6P enables Layer-2 peers to allocate, move or deallocate cells in their respective schedules to communicate. 6P operates at the 6top layer.

6P Transaction: A 2-way or 3-way sequence of 6P messages used by Layer-2 peers to modify their communication schedule.

ASN (Absolute Slot Number): Defined in [IEEE802154], the ASN is the total number of timeslots that have elapsed since the Epoch Time when the TSCH network started. Incremented by one at each timeslot. It is wide enough to not roll over in practice.

bundle: A group of equivalent scheduled cells, i.e., cells identified by different [slotOffset, channelOffset], which are scheduled for a same purpose, with the same neighbor, with the same flags, and the same slotframe. The size of the bundle refers to the number of cells it contains. For a given slotframe length, the size of the bundle translates directly into bandwidth. A bundle is a local abstraction that represents a half-duplex link for either sending or receiving, with bandwidth that amounts to the sum of the cells in the bundle.

Layer-2 vs. Layer-3 bundle: Bundles are associated for either Layer-2 (switching) or Layer-3 (routing) forwarding operations. A pair of Layer-3 bundles (one for each direction) maps to an IP Link with a neighbor, whereas a set of Layer-2 bundles (of an "arbitrary" cardinality and direction) corresponds to the relation of one or more incoming bundle(s) from the previous-hop neighbor(s) with one or more outgoing bundle(s) to the next-hop neighbor(s) along a Track as part of the switching role, which may include replication and elimination.

CCA (Clear Channel Assessment): A mechanism defined in [IEEE802154] whereby nodes listen to the channel before sending to detect ongoing transmissions from other parties. Because the network is synchronized, CCA cannot be used to detect colliding transmissions within the same network, but it can be used to detect other radio networks in vicinity.

cell: A unit of transmission resource in the CDU matrix, a cell is identified by a slotOffset and a channelOffset. A cell can be scheduled or unscheduled.

Channel Distribution/Usage (CDU) matrix: : A matrix of cells (i,j) representing the spectrum (channel) distribution among the different nodes in the 6TiSCH network. The CDU matrix has width in timeslots, equal to the period of the network scheduling operation, and height equal to the number of available channels. Every cell (i,j) in the CDU, identified by (slotOffset, channelOffset), belongs to a specific chunk.

channelOffset: Identifies a row in the TSCH schedule. The number of channelOffset values is bounded by the number of available frequencies. The channelOffset translates into a frequency with a function that depends on the absolute time when the communication takes place, resulting in a channel hopping operation.

chunk: A well-known list of cells, distributed in time and

frequency, within a CDU matrix. A chunk represents a portion of a CDU matrix. The partition of the CDU matrix in chunks is globally known by all the nodes in the network to support the appropriation process, which is a negotiation between nodes within an interference domain. A node that manages to appropriate a chunk gets to decide which transmissions will occur over the cells in the chunk within its interference domain, i.e., a parent node will decide when the cells within the appropriated chunk are used and by which node, among its children.

CoJP (Constrained Join Protocol): The Constrained Join Protocol (CoJP) enables a pledge to securely join a 6TiSCH network and obtain network parameters over a secure channel. Minimal Security Framework for 6TiSCH [MIN-SECURITY] defines the minimal CoJP setup with pre-shared keys defined. In that mode, CoJP can operate with a single round trip exchange.

dedicated cell: A cell that is reserved for a given node to transmit to a specific neighbor.

deterministic network: The generic concept of deterministic network is defined in the "DetNet Architecture" [RFC8655] document. When applied to 6TiSCH, it refers to the reservation of Tracks which guarantees an end-to-end latency and optimizes the Packet Delivery Ratio (PDR) for well-characterized flows.

distributed cell reservation: A reservation of a cell done by one or more in-network entities.

distributed Track reservation: A reservation of a Track done by one or more in-network entities.

EB (Enhanced Beacon): A special frame defined in [IEEE802154] used by a node, including the JP, to announce the presence of the network. It contains enough information for a pledge to synchronize to the network.

hard cell: A scheduled cell which the 6top sublayer may not relocate.

hopping sequence: Ordered sequence of frequencies, identified by a Hopping_Sequence_ID, used for channel hopping when translating the channelOffset value into a frequency.

IE (Information Element): Type-Length-Value containers placed at the

end of the MAC header, used to pass data between layers or devices. Some IE identifiers are managed by the IEEE [IEEE802154]. Some IE identifiers are managed by the IETF [RFC8137], and [ENH-BEACON] uses one subtype to support the selection of the Join Proxy.

join process: The overall process that includes the discovery of the network by pledge(s) and the execution of the join protocol.

join protocol: The protocol that allows the pledge to join the network. The join protocol encompasses authentication, authorization and parameter distribution. The join protocol is executed between the pledge and the JRC.

joined node: The new device, after having completed the join process, often just called a node.

JP (Join Proxy): Node already part of the 6TiSCH network that serves as a relay to provide connectivity between the pledge and the JRC. The JP announces the presence of the network by regularly sending EB frames.

JRC (Join Registrar/Coordinator): Central entity responsible for the authentication, authorization and configuration of the pledge.

link: A communication facility or medium over which nodes can communicate at the Link-Layer, the layer immediately below IP. In 6TiSCH, the concept is implemented as a collection of Layer-3 bundles. Note: the IETF parlance for the term "Link" is adopted, as opposed to the IEEE Std. 802.15.4 terminology.

Operational Technology: OT refers to technology used in automation, for instance in industrial control networks. The convergence of IT and OT is the main object of the Industrial Internet of Things (IIOT).

pledge: A new device that attempts to join a 6TiSCH network.

(to) relocate a cell: The action operated by the 6top sublayer of changing the slotOffset and/or channelOffset of a soft cell.

(to) schedule a cell: The action of turning an unscheduled cell into a scheduled cell.

scheduled cell: A cell which is assigned a neighbor MAC address

(broadcast address is also possible), and one or more of the following flags: TX, RX, Shared and Timekeeping. A scheduled cell can be used by the IEEE Std. 802.15.4 TSCH implementation to communicate. A scheduled cell can either be a hard or a soft cell.

SF (6top Scheduling Function): The cell management entity that adds or deletes cells dynamically based on application networking requirements. The cell negotiation with a neighbor is done using 6P.

SFID (6top Scheduling Function Identifier): A 4-bit field identifying an SF.

shared cell: A cell marked with both the "TX" and "shared" flags. This cell can be used by more than one transmitter node. A back-off algorithm is used to resolve contention.

slotframe: A collection of timeslots repeating in time, analogous to a superframe in that it defines periods of communication opportunities. It is characterized by a slotframe_ID, and a slotframe_size. Multiple slotframes can coexist in a node's schedule, i.e., a node can have multiple activities scheduled in different slotframes, based on the priority of its packets/traffic flows. The timeslots in the Slotframe are indexed by the SlotOffset; the first timeslot is at SlotOffset 0.

slotOffset: A column in the TSCH schedule, i.e., the number of timeslots since the beginning of the current iteration of the slotframe.

soft cell: A scheduled cell which the 6top sublayer can relocate.

time source neighbor: A neighbor that a node uses as its time reference, and to which it needs to keep its clock synchronized.

timeslot: A basic communication unit in TSCH which allows a transmitter node to send a frame to a receiver neighbor, and that receiver neighbor to optionally send back an acknowledgment.

Track: A Track is a Directed Acyclic Graph (DAG) that is used as a

complex multi-hop path to the destination(s) of the path. In the case of unicast traffic, the Track is a Destination Oriented DAG (DODAG) where the Root of the DODAG is the destination of the unicast traffic. A Track enables replication, elimination and reordering functions on the way (more on those functions in [RFC8655]. A Track reservation locks physical resources such as cells and buffers in every node along the DODAG. A Track is associated with a owner that can be for instance the destination of the Track.

TrackID: A TrackID is either globally unique, or locally unique to the Track owner, in which case the identification of the owner must be provided together with the TrackID to provide a full reference to the Track. typically, the Track owner is the ingress of the Track then the IPv6 source address of packets along the Track can be used as identification of the owner and a local InstanceID [RFC6550] in the namespace of that owner can be used as TrackID. If the Track is reversible, then the owner is found in the IPv6 destination address of a packet coming back along the Track. In that case, a RPL Packet Information [RFC6550] in an IPv6 packet can unambiguously identify the Track and can be expressed in a compressed form using [RFC8138].

TSCH: A medium access mode of the IEEE Std. 802.15.4 [IEEE802154] standard which uses time synchronization to achieve ultra-low-power operation, and channel hopping to enable high reliability.

TSCH Schedule: A matrix of cells, each cell indexed by a slotOffset and a channelOffset. The TSCH schedule contains all the scheduled cells from all slotframes and is sufficient to qualify the communication in the TSCH network. The number of channelOffset values (the "height" of the matrix) is equal to the number of available frequencies.

Unscheduled Cell: A cell which is not used by the IEEE Std. 802.15.4 TSCH implementation.

2.2. Abbreviations

This document uses the following abbreviations:

6BBR: 6LoWPAN Backbone Router (router with a proxy ND function)

6LBR: 6LoWPAN Border Router (authoritative on DAD)

6LN: 6LoWPAN Node

6LR: 6LoWPAN Router (relay to the registration process)

6CIO: Capability Indication Option

(E)ARO: (Extended) Address Registration Option

(E)DAR: (Extended) Duplicate Address Request

(E)DAC: (Extended) Duplicate Address Confirmation

DAD: Duplicate Address Detection

DODAG: Destination-Oriented Directed Acyclic Graph

LLN: Low-Power and Lossy Network (a typical IoT network)

NA: Neighbor Advertisement

NCE: Neighbor Cache Entry

ND: Neighbor Discovery

NDP: Neighbor Discovery Protocol

PCE: Path Computation Element

NME: Network Management Entity

ROVR: Registration Ownership Verifier (pronounced rover)

RPL: IPv6 Routing Protocol for LLNs (pronounced ripple)

RA: Router Advertisement

RS: Router Solicitation

TSCH: timeslotted Channel Hopping

TID: Transaction ID (a sequence counter in the EARO)

2.3. Related Documents

The draft also conforms to the terms and models described in [RFC3444] and [RFC5889] and uses the vocabulary and the concepts defined in [RFC4291] for the IPv6 Architecture and refers [RFC4080] for reservation

The draft uses domain-specific terminology defined or referenced in:

6LoWPAN ND "Neighbor Discovery Optimization for Low-power and Lossy Networks" [RFC6775] and "Registration Extensions for 6LoWPAN Neighbor Discovery" [RFC8505],

"Terms Used in Routing for Low-Power and Lossy Networks (LLNs)" [RFC7102],

and RPL "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)" [RFC6552], and "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550].

Other terms in use in LLNs are found in "Terminology for Constrained-Node Networks" [RFC7228].

Readers are expected to be familiar with all the terms and concepts that are discussed in

* "Neighbor Discovery for IP version 6" [RFC4861], and "IPv6 Stateless Address Autoconfiguration" [RFC4862].

In addition, readers would benefit from reading:

* "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing" [RFC6606],

* "Multi-Link Subnet Issues" [RFC4903], and

* "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919]

prior to this specification for a clear understanding of the art in ND-proxying and binding.

3. High Level Architecture

3.1. A Non-Broadcast Multi-Access Radio Mesh Network

A 6TiSCH network is an IPv6 [RFC8200] subnet which, in its basic configuration illustrated in Figure 1, is a single Low-Power Lossy Network (LLN) operating over a synchronized TSCH-based mesh.

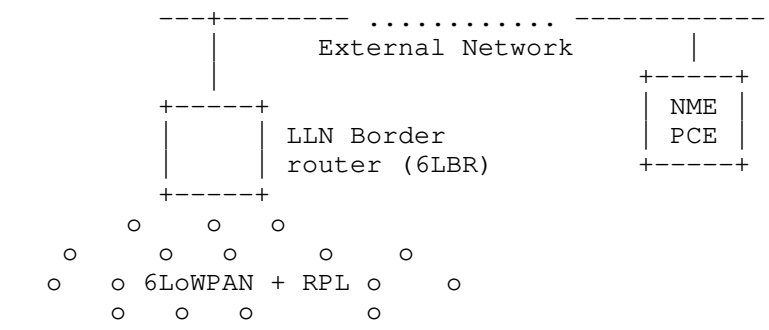

Figure 1: Basic Configuration of a 6TiSCH Network

Inside a 6TiSCH LLN, nodes rely on 6LoWPAN Header Compression (6LoWPAN HC) [RFC6282] to encode IPv6 packets. From the perspective of the network layer, a single LLN interface (typically an IEEE Std. 802.15.4-compliant radio) may be seen as a collection of Links with different capabilities for unicast or multicast services.

6TiSCH nodes join a mesh network by attaching to nodes that are already members of the mesh (see Section 4.2.1). The security aspects of the join process are further detailed in Section 6. In a mesh network, 6TiSCH nodes are not necessarily reachable from one another at Layer-2 and an LLN may span over multiple links.

This forms a homogeneous non-broadcast multi-access (NBMA) subnet, which is beyond the scope of IPv6 Neighbor Discovery (IPv6 ND) [RFC4861][RFC4862]. 6LoWPAN Neighbor Discovery (6LoWPAN ND) [RFC6775][RFC8505] specifies extensions to IPv6 ND that enable ND operations in this type of subnet that can be protected against address theft and impersonation with [AP-ND].

Once it has joined the 6TiSCH network, a node acquires IPv6 Addresses and register them using 6LoWPAN ND. This guarantees that the addresses are unique and protects the address ownership over the subnet, more in Section 4.2.2.

Within the NBMA subnet, RPL [RFC6550] enables routing in the so-called Route Over fashion, either in storing (stateful) or non-storing (stateless, with routing headers) mode. From there, some nodes can act as routers for 6LoWPAN ND and RPL operations, as detailed in Section 4.1.

With TSCH, devices are time-synchronized at the MAC level. The use of a particular RPL Instance for time synchronization is discussed in Section 4.3.4. With this mechanism, the time synchronization starts at the RPL Root and follows the RPL loopless routing topology.

RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) within Instances of the protocol, each Instance being associated with an Objective Function (OF) to form a routing topology. A particular 6TiSCH node, the LLN Border Router (6LBR), acts as RPL Root, 6LoWPAN HC terminator, and Border Router for the LLN to the outside. The 6LBR is usually powered. More on RPL Instances can be found in section 3.1 of RPL [RFC6550], in particular "3.1.2. RPL Identifiers" and "3.1.3. Instances, DODAGs, and DODAG Versions". RPL adds artifacts in the data packets that are compressed with a 6LoWPAN addition 6LoRH [RFC8138].

Additional routing and scheduling protocols may be deployed to establish on-demand Peer-to-Peer routes with particular characteristics inside the 6TiSCH network. This may be achieved in a centralized fashion by a Path Computation Element (PCE) [PCE] that programs both the routes and the schedules inside the 6TiSCH nodes, or by in a distributed fashion using a reactive routing protocol and a Hop-by-Hop scheduling protocol.

This architecture expects that a 6LoWPAN node can connect as a leaf to a RPL network, where the leaf support is the minimal functionality to connect as a host to a RPL network without the need to participate to the full routing protocol. The architecture also expects that a 6LoWPAN node that is not aware at all of the RPL protocol may also connect as described in [RUL-DRAFT].

3.2. A Multi-Link Subnet Model

An extended configuration of the subnet comprises multiple LLNs as illustrated in Figure 2. In the extended configuration, a Routing Registrar [RFC8505] may be connected to the node that acts as RPL Root and / or 6LoWPAN 6LBR and provides connectivity to the larger campus / factory plant network over a high-speed backbone or a back-haul link. The Routing registrar may perform IPv6 ND proxy operations, or redistribute the registration in a routing protocol such as OSPF [RFC5340] or BGP [RFC2545], or inject a route in a mobility protocol such as MIPv6 [RFC6275], NEMO [RFC3963], or LISP [RFC6830].

Multiple LLNs can be interconnected and possibly synchronized over a backbone, which can be wired or wireless. The backbone can operate with IPv6 ND [RFC4861][RFC4862] procedures or an hybrid of IPv6 ND and 6LoWPAN ND [RFC6775][RFC8505][AP-ND].

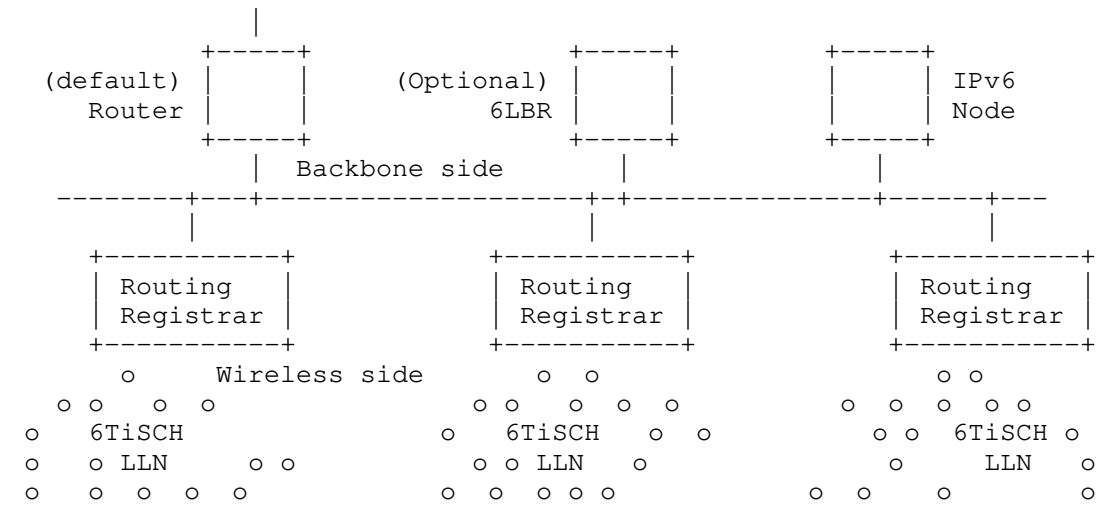

Figure 2: Extended Configuration of a 6TiSCH Network

A Routing Registrar that performs proxy IPv6 ND operations over the backbone on behalf of the 6TiSCH nodes is called a Backbone Router (6BBR) [6BBR-DRAFT]. The 6BBRs are placed along the wireless edge of a Backbone, and federate multiple wireless links to form a single MultiLink Subnet. The 6BBRs synchronize with one another over the backbone, so as to ensure that the multiple LLNs that form the IPv6 subnet stay tightly synchronized.

The use of multicast can also be reduced on the backbone with a registrar that would contribute to Duplicate Address Detection as well as Address Lookup using only unicast request/response exchanges. [I-D.thubert-6man-unicast-lookup] is a proposed method that presents an example of how to this could be achieved with an extension of [RFC8505], using an optional 6LBR as a SubNet-level registrar, as illustrated in Figure 2.

As detailed in Section 4.1 the 6LBR that serves the LLN and the Root of the RPL network need to share information about the devices that are learned through either 6LoWPAN ND or RPL but not both. The preferred way of achieving this is to collocate/combine them. The combined RPL Root and 6LBR may be collocated with the 6BBR, or directly attached to the 6BBR. In the latter case, it leverages the extended registration process defined in [RFC8505] to proxy the 6LoWPAN ND registration to the 6BBR on behalf of the LLN nodes, so that the 6BBR may in turn perform proxy classical ND operations over the backbone.

The DetNet Architecture [RFC8655] studies Layer-3 aspects of Deterministic Networks, and covers networks that span multiple Layer-2 domains. If the Backbone is Deterministic (such as defined by the Time Sensitive Networking WG at IEEE), then the Backbone Router ensures that the end-to-end deterministic behavior is maintained between the LLN and the backbone.

3.3. TSCH: A Deterministic MAC Layer

Though at a different time scale (several orders of magnitude), both IEEE Std. 802.1TSN and IEEE Std. 802.15.4 TSCH standards provide Deterministic capabilities to the point that a packet that pertains to a certain flow may traverse a network from node to node following a precise schedule, as a train that enters and then leaves intermediate stations at precise times along its path.

With TSCH, time is formatted into timeslots, and individual communication cells are allocated to unicast or broadcast communication at the MAC level. The time-slotted operation reduces collisions, saves energy, and enables to more closely engineer the network for deterministic properties. The channel hopping aspect is a simple and efficient technique to combat multipath fading and co-channel interference.

6TiSCH builds on the IEEE Std. 802.15.4 TSCH MAC and inherits its advanced capabilities to enable them in multiple environments where they can be leveraged to improve automated operations. The 6TiSCH Architecture also inherits the capability to perform a centralized route computation to achieve deterministic properties, though it relies on the IETF DetNet Architecture [RFC8655], and IETF components such as the PCE [PCE], for the protocol aspects.

On top of this inheritance, 6TiSCH adds capabilities for distributed routing and scheduling operations based on the RPL routing protocol and capabilities to negotiate schedule adjustments between peers. These distributed routing and scheduling operations simplify the deployment of TSCH networks and enable wireless solutions in a larger variety of use cases from operational technology in general. Examples of such use-cases in industrial environments include plant setup and decommissioning, as well as monitoring of lots of lesser importance measurements such as corrosion and events and mobile workers accessing local devices.

3.4. Scheduling TSCH

A scheduling operation attributes cells in a Time-Division-Multiplexing (TDM) / Frequency-Division Multiplexing (FDM) matrix called the Channel distribution/usage (CDU) to either individual transmissions or as multi-access shared resources. The CDU matrix can be formatted in chunks that can be allocated exclusively to particular nodes to enable distributed scheduling without collision. More in Section 4.3.5.

From the standpoint of a 6TiSCH node (at the MAC layer), its schedule is the collection of the timeslots at which it must wake up for transmission, and the channels to which it should either send or listen at those times. The schedule is expressed as one or more slotframes that repeat over and over. Slotframes may collide and require a device to wake up at a same time, in which case the slotframe with the highest priority is actionable.

The 6top sublayer (see Section 4.3 for more) hides the complexity of the schedule from the upper layers. The Link abstraction that IP traffic utilizes is composed of a pair of Layer-3 cell bundles, one to receive and one to transmit. Some of the cells may be shared, in which case the 6top sublayer must perform some arbitration.

Scheduling enables multiple communications at a same time in a same interference domain using different channels; but a node equipped with a single radio can only either transmit or receive on one channel at any point of time. Scheduled cells that fulfil the same role, e.g., receive IP packets from a peer, are grouped in bundles.

The 6TiSCH architecture identifies four ways a schedule can be managed and CDU cells can be allocated: Static Scheduling, Neighbor-to-Neighbor Scheduling, Centralized (or Remote) Monitoring and Schedule Management, and Hop-by-hop Scheduling.

Static Scheduling: This refers to the minimal 6TiSCH operation whereby a static schedule is configured for the whole network for use in a Slotted ALOHA [S-ALOHA] fashion. The static schedule is distributed through the native methods in the TSCH MAC layer and does not preclude other scheduling operations to co-exist on a same 6TiSCH network. A static schedule is necessary for basic operations such as the join process and for interoperability during the network formation, which is specified as part of the Minimal 6TiSCH Configuration [RFC8180].

Neighbor-to-Neighbor Scheduling: This refers to the dynamic

adaptation of the bandwidth of the Links that are used for IPv6 traffic between adjacent peers. Scheduling Functions such as the "6TiSCH Minimal Scheduling Function (MSF)" [MSF] influence the operation of the MAC layer to add, update and remove cells in its own, and its peer's schedules using 6P [RFC8480], for the negotiation of the MAC resources.

Centralized (or Remote) Monitoring and Schedule Management: This refers to the central computation of a schedule and the capability to forward a frame based on the cell of arrival. In that case, the related portion of the device schedule as well as other device resources are managed by an abstract Network Management Entity (NME), which may cooperate with the PCE to minimize the interaction with and the load on the constrained device. This model is the TSCH adaption of the DetNet Architecture [RFC8655], and it enables Traffic Engineering with deterministic properties.

Hop-by-hop Scheduling: This refers to the possibility to reserves cells along a path for a particular flow using a distributed mechanism.

It is not expected that all use cases will require all those mechanisms. Static Scheduling with minimal configuration one is the only one that is expected in all implementations, since it provides a simple and solid basis for convergecast routing and time distribution.

A deeper dive in those mechanisms can be found in Section 4.4.

3.5. Distributed vs. Centralized Routing

6TiSCH enables a mixed model of centralized routes and distributed routes. Centralized routes can for example be computed by an entity such as a PCE. 6TiSCH leverages the RPL [RFC6550] routing protocol for interoperable distributed routing operations.

Both methods may inject routes in the Routing Tables of the 6TiSCH routers. In either case, each route is associated with a 6TiSCH topology that can be a RPL Instance topology or a Track. The 6TiSCH topology is indexed by a RPLInstanceID, in a format that reuses the RPLInstanceID as defined in RPL.

RPL [RFC6550] is applicable to Static Scheduling and Neighbor-to-Neighbor Scheduling. The architecture also supports a centralized routing model for Remote Monitoring and Schedule Management. It is expected that a routing protocol that is more optimized for point-to-point routing than RPL [RFC6550], such as the Asymmetric AODV-P2P-RPL

in Low-Power and Lossy Networks" [I-D.ietf-roll-aodv-rpl] AODV-RPL), which derives from the Ad Hoc On-demand Distance Vector Routing (AODV) [I-D.ietf-manet-aodvv2] will be selected for Hop-by-hop Scheduling.

Both RPL and PCE rely on shared sources such as policies to define Global and Local RPLInstanceIDs that can be used by either method. It is possible for centralized and distributed routing to share a same topology. Generally they will operate in different slotframes, and centralized routes will be used for scheduled traffic and will have precedence over distributed routes in case of conflict between the slotframes.

3.6. Forwarding Over TSCH

The 6TiSCH architecture supports three different forwarding models. One is the classical IPv6 Forwarding, where the node selects a feasible successor at Layer-3 on a per packet basis and based on its routing table. The second derives from Generic MPLS (G-MPLS) for so-called Track Forwarding, whereby a frame received at a particular timeslot can be switched into another timeslot at Layer-2 without regard to the upper layer protocol. The third model is the 6LoWPAN Fragment Forwarding, which allows to forward individual 6loWPAN fragments along a route that is setup by the first fragment.

In more details:

IPv6 Forwarding: This is the classical IP forwarding model, with a Routing Information Based (RIB) that is installed by the RPL routing protocol and used to select a feasible successor per packet. The packet is placed on an outgoing Link, that the 6top layer maps into a (Layer-3) bundle of cells, and scheduled for transmission based on QoS parameters. Besides RPL, this model also applies to any routing protocol which may be operated in the 6TiSCH network, and corresponds to all the distributed scheduling models, Static, Neighbor-to-Neighbor and Hop-by-Hop Scheduling.

G-MPLS Track Forwarding: This model corresponds to the Remote Monitoring and Schedule Management. In this model, a central controller (hosting a PCE) computes and installs the schedules in the devices per flow. The incoming (Layer-2) bundle of cells from the previous node along the path determines the outgoing (Layer-2) bundle towards the next hop for that flow as determined by the PCE. The programmed sequence for bundles is called a Track and can assume DAG shapes that are more complex than a simple direct sequence of nodes.

6LoWPAN Fragment Forwarding: This is a hybrid model that derives

from IPv6 forwarding for the case where packets must be fragmented at the 6LoWPAN sublayer. The first fragment is forwarded like any IPv6 packet and leaves a state in the intermediate hops to enable forwarding of the next fragments that do not have a IP header without the need to recompose the packet at every hop.

A deeper dive on these operations can be found in Section 4.6.

The following table summarizes how the forwarding models apply to the various routing and scheduling possibilities:

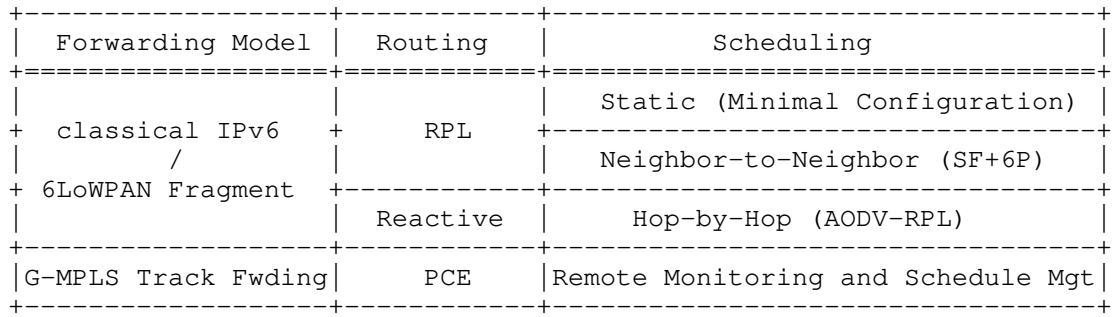

Figure 3

3.7. 6TiSCH Stack

The IETF proposes multiple techniques for implementing functions related to routing, transport or security.

The 6TiSCH architecture limits the possible variations of the stack and recommends a number of base elements for LLN applications to control the complexity of possible deployments and device interactions, and to limit the size of the resulting object code. In particular, UDP [RFC0768], IPv6 [RFC8200] and the Constrained Application Protocol [RFC7252] (CoAP) are used as the transport / binding of choice for applications and management as opposed to TCP and HTTP.

The resulting protocol stack is represented in Figure 4:

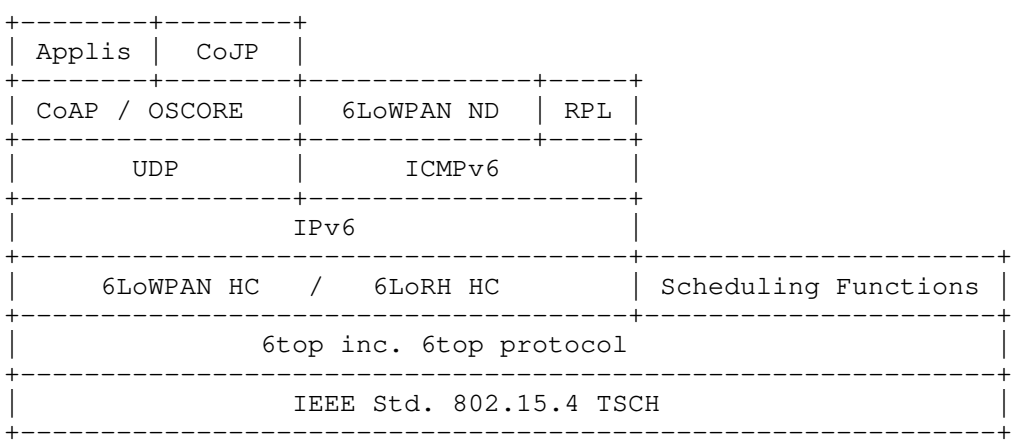

Figure 4: 6TiSCH Protocol Stack

RPL is the routing protocol of choice for LLNs. So far, there was no identified need to define a 6TiSCH specific Objective Function. The Minimal 6TiSCH Configuration [RFC8180] describes the operation of RPL over a static schedule used in a Slotted ALOHA fashion [S-ALOHA], whereby all active slots may be used for emission or reception of both unicast and multicast frames.

The 6LoWPAN Header Compression [RFC6282] is used to compress the IPv6 and UDP headers, whereas the 6LoWPAN Routing Header (6LoRH) [RFC8138] is used to compress the RPL artifacts in the IPv6 data packets, including the RPL Packet Information (RPI), the IP-in-IP encapsulation to/from the RPL Root, and the Source Route Header (SRH) in non-storing mode. "When to use RFC 6553, 6554 and IPv6-in-IPv6" [USEofRPLinfo] provides the details on when headers or encapsulation are needed.

The Object Security for Constrained RESTful Environments (OSCORE) [I-D.ietf-core-object-security], is leveraged by the Constrained Join Protocol (CoJP) and is expected to be the primary protocol for the protection of the application payload as well. The application payload may also be protected by the Datagram Transport Layer Security (DTLS) [RFC6347] sitting either under CoAP or over CoAP so it can traverse proxies.

The 6TiSCH Operation sublayer (6top) is a sublayer of a Logical Link Control (LLC) that provides the abstraction of an IP link over a TSCH MAC and schedules packets over TSCH cells, as further discussed in the next sections, providing in particular dynamic cell allocation with the 6top Protocol (6P) [RFC8480].

The reference stack presented in this document was implemented and interop-tested by a conjunction of opensource, IETF and ETSI efforts. One goal is to help other bodies to adopt the stack as a whole, making the effort to move to an IPv6-based IoT stack easier.

For a particular environment, some of the choices that are made in this architecture may not be relevant. For instance, RPL is not required for star topologies and mesh-under Layer-2 routed networks, and the 6LoWPAN compression may not be sufficient for ultra-constrained cases such as some Low-Power Wide Area (LPWA) networks. In such cases, it is perfectly doable to adopt a subset of the selection that is presented hereafter and then select alternate components to complete the solution wherever needed.

3.8. Communication Paradigms and Interaction Models

Section 2.1 provides the terms of Communication Paradigms and Interaction Models, in relation with "On the Difference between Information Models and Data Models" [RFC3444]. A Communication Paradigm would be an abstract view of a protocol exchange, and would come with an Information Model for the information that is being exchanged. In contrast, an Interaction Model would be more refined and could point to standard operation such as a Representational state transfer (REST) "GET" operation and would match a Data Model for the data that is provided over the protocol exchange.

Section 2.1.3 of [I-D.ietf-roll-rpl-industrial-applicability] and next sections discuss application-layer paradigms, such as Source-sink (SS) that is a Multipeer to Multipeer (MP2MP) model primarily used for alarms and alerts, Publish-subscribe (PS, or pub/sub) that is typically used for sensor data, as well as Peer-to-peer (P2P) and Peer-to-multipeer (P2MP) communications.

Additional considerations on Duocast - one sender, two receivers for redundancy - and its N-cast generalization are also provided. Those paradigms are frequently used in industrial automation, which is a major use case for IEEE Std. 802.15.4 TSCH wireless networks with [ISA100.11a] and [WirelessHART], that provides a wireless access to [HART] applications and devices.

This document focuses on Communication Paradigms and Interaction Models for packet forwarding and TSCH resources (cells) management. Management mechanisms for the TSCH schedule at Link-Layer (one-hop), Network-layer (multihop along a Track), and Application-layer (remote control) are discussed in Section 4.4. Link-Layer frame forwarding interactions are discussed in Section 4.6, and Network-layer Packet routing is addressed in Section 4.7.

4. Architecture Components

4.1. 6LoWPAN (and RPL)

A RPL DODAG is formed of a Root, a collection of routers, and leaves that are hosts. Hosts are nodes which do not forward packets that they did not generate. RPL-aware leaves will participate to RPL to advertise their own addresses, whereas RPL-unaware leaves depend on a connected RPL router to do so. RPL interacts with 6LoWPAN ND at multiple levels, in particular at the Root and in the RPL-unaware leaves.

4.1.1. RPL-Unaware Leaves and 6LoWPAN ND

RPL needs a set of information to advertise a leaf node through a Destination Advertisement Object (DAO) message and establish reachability.

"Routing for RPL Leaves" [RUL-DRAFT] details the basic interaction of 6LoWPAN ND and RPL and enables a plain 6LN that supports [RFC8505] to obtain return connectivity via the RPL network as an RPL-unaware leaf. The leaf indicates that it requires reachability services for the Registered Address from a Routing Registrar by setting a 'R' flag in the Extended Address Registration Option [RFC8505], and it provides a TID that maps to a sequence number in section 7 of RPL [RFC6550].

[RUL-DRAFT] also enables the leaf to signal the RPL InstanceID that it wants to participate to using the Opaque field of the EARO. On the backbone, the InstanceID is expected to be mapped to an overlay that matches the RPL Instance, e.g., a Virtual LAN (VLAN) or a virtual routing and forwarding (VRF) instance.

Though at the time of this writing the above specification enables a model where the separation is possible, this architecture recommends to collocate the functions of 6LBR and RPL Root.

4.1.2. 6LBR and RPL Root

With the 6LowPAN ND [RFC6775], information on the 6LBR is disseminated via an Authoritative Border Router Option (ABRO) in RA messages. [RFC8505] extends [RFC6775] to enable a registration for routing and proxy ND. The capability to support [RFC8505] is indicated in the 6LoWPAN Capability Indication Option (6CIO). The discovery and liveliness of the RPL Root are obtained through RPL [RFC6550] itself.

When 6LoWPAN ND is coupled with RPL, the 6LBR and RPL Root functionalities are co-located in order that the address of the 6LBR be indicated by RPL DIO messages and to associate the unique ID from the EDAR/EDAC [RFC8505] exchange with the state that is maintained by RPL.

Section 7 of [RUL-DRAFT] specifies how the DAO messages are used to reconfirm the registration, thus eliminating a duplication of functionality between DAO and EDAR/EDAC messages, as illustrated in Figure 7. [RUL-DRAFT] also provides the protocol elements that are needed when the 6LBR and RPL Root functionalities are not co-located.

Even though the Root of the RPL network is integrated with the 6LBR, it is logically separated from the Backbone Router (6BBR) that is used to connect the 6TiSCH LLN to the backbone. This way, the Root has all information from 6LoWPAN ND and RPL about the LLN devices attached to it.

This architecture also expects that the Root of the RPL network (proxy-)registers the 6TiSCH nodes on their behalf to the 6BBR, for whatever operation the 6BBR performs on the backbone, such as ND proxy, or redistribution in a routing protocol. This relies on an extension of the 6LoWPAN ND registration described in [6BBR-DRAFT].

This model supports the movement of a 6TiSCH device across the Multi-Link Subnet, and allows the proxy registration of 6TiSCH nodes deep into the 6TiSCH LLN by the 6LBR / RPL Root. This is why in [RFC8505] the Registered Address is signaled in the Target Address field of the NS message as opposed to the IPv6 Source Address, which, in the case of a proxy registration, is that of the 6LBR / RPL Root itself.

4.2. Network Access and Addressing

4.2.1. Join Process

A new device, called the pledge, undergoes the join protocol to become a node in a 6TiSCH network. This usually occurs only once when the device is first powered on. The pledge communicates with the Join Registrar/Coordinator (JRC) of the network through a Join Proxy (JP), a radio neighbor of the pledge.

The JP is discovered though MAC layer beacons. When multiple JPs from possibly multiple networks are visible, trial and error till an acceptable position in the right network is obtained becomes ineffficient. [ENH-BEACON] adds a new subtype in the Information Element that was delegated to the IETF [RFC8137] and provides visibility on the network that can be joined and the willingness by the JP and the Root to be used by the pledge.

The join protocol provides the following functionality:

* Mutual authentication

* Authorization

* Parameter distribution to the pledge over a secure channel

Minimal Security Framework for 6TiSCH [MIN-SECURITY] defines the minimal mechanisms required for this join process to occur in a secure manner. The specification defines the Constrained Join Protocol (CoJP) that is used to distribute the parameters to the pledge over a secure session established through OSCORE [I-D.ietf-core-object-security], and a secure configuration of the network stack. In the minimal setting with pre-shared keys (PSKs), CoJP allows the pledge to join after a single round-trip exchange with the JRC. The provisioning of the PSK to the pledge and the JRC needs to be done out of band, through a 'one-touch' bootstrapping process, which effectively enrolls the pledge into the domain managed by the JRC.

In certain use cases, the 'one touch' bootstrapping is not feasible due to the operational constraints and the enrollment of the pledge into the domain needs to occur in-band. This is handled through a 'zero-touch' extension of the Minimal Security Framework for 6TiSCH. Zero touch [I-D.ietf-6tisch-dtsecurity-zerotouch-join] extension leverages the 'Bootstrapping Remote Secure Key Infrastructures (BRSKI)' [[I-D.ietf-anima-bootstrapping-keyinfra] work to establish a shared secret between a pledge and the JRC without necessarily having them belong to a common (security) domain at join time. This happens through inter-domain communication occurring between the JRC of the

network and the domain of the pledge, represented by a fourth entity, Manufacturer Authorized Signing Authority (MASA). Once the zero-touch exchange completes, the CoJP exchange defined in [MIN-SECURITY] is carried over the secure session established between the pledge and the JRC.

Figure 5 depicts the join process and where a Link-Local Address (LLA) is used, versus a Global Unicast Address (GUA).

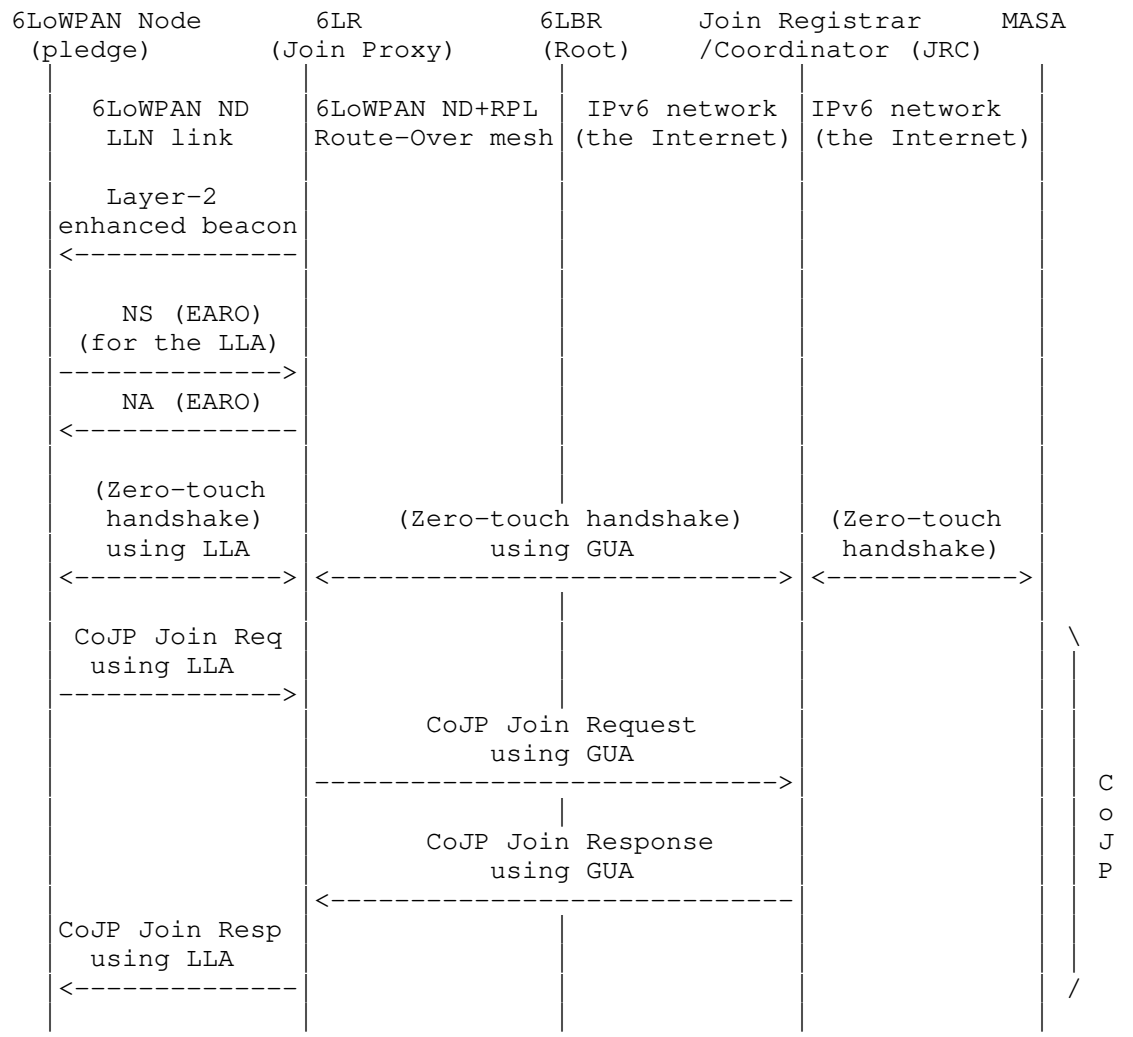

Figure 5: Join process in a Multi-Link Subnet. Parentheses () denote optional exchanges.

4.2.2. Registration

Once the pledge successfully completes the CoJP protocol and becomes a network node, it obtains the network prefix from neighboring routers and registers its IPv6 addresses. As detailed in Section 4.1, the combined 6LoWPAN ND 6LBR and Root of the RPL network learn information such as the device Unique ID (from 6LoWPAN ND) and the updated Sequence Number (from RPL), and perform 6LoWPAN ND proxy registration to the 6BBR of behalf of the LLN nodes.

Figure 6 illustrates the initial IPv6 signaling that enables a 6LN to form a global address and register it to a 6LBR using 6LoWPAN ND [RFC8505], is then carried over RPL to the RPL Root, and then to the 6BBR. This flow happens just once when the address is created and first registered.

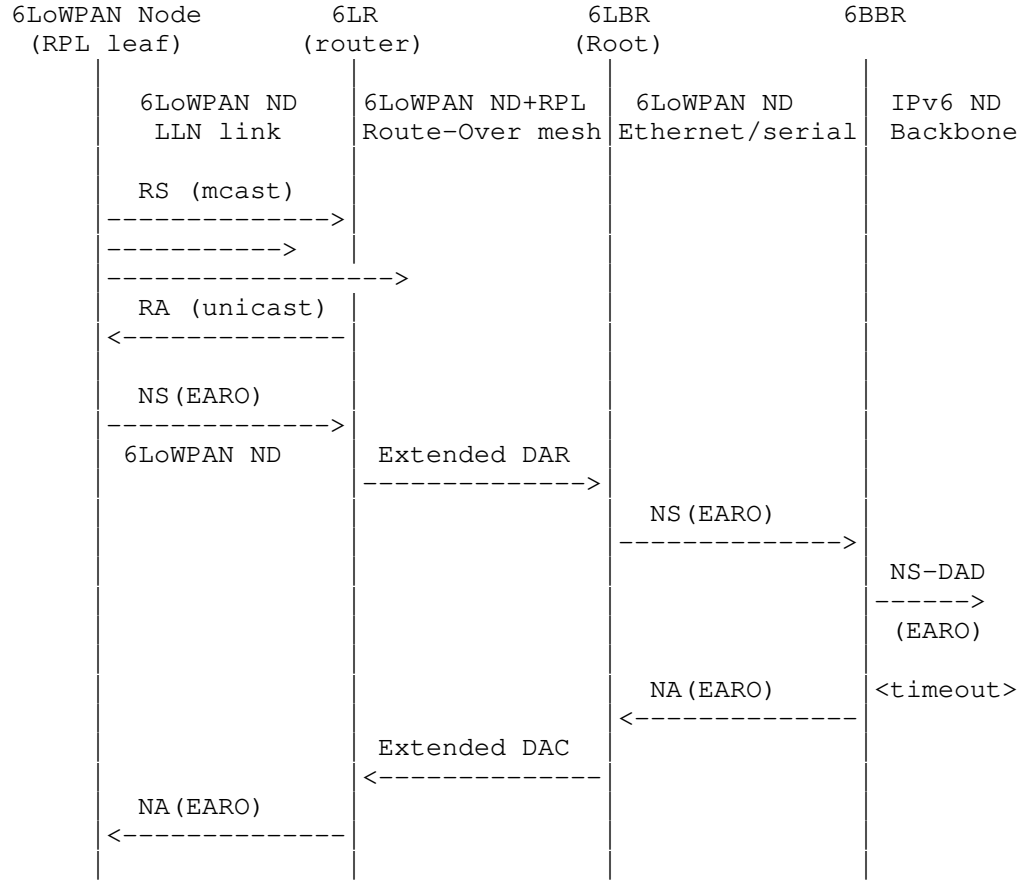

Figure 6: Initial Registration Flow over Multi-Link Subnet

Figure 7 illustrates the repeating IPv6 signaling that enables a 6LN to keep a global address alive and registered to its 6LBR using 6LoWPAN ND to the 6LR, RPL to the RPL Root, and then 6LoWPAN ND again to the 6BBR, which avoids repeating the Extended DAR/DAC flow across the network when RPL can suffice as a keep-alive mechanism.

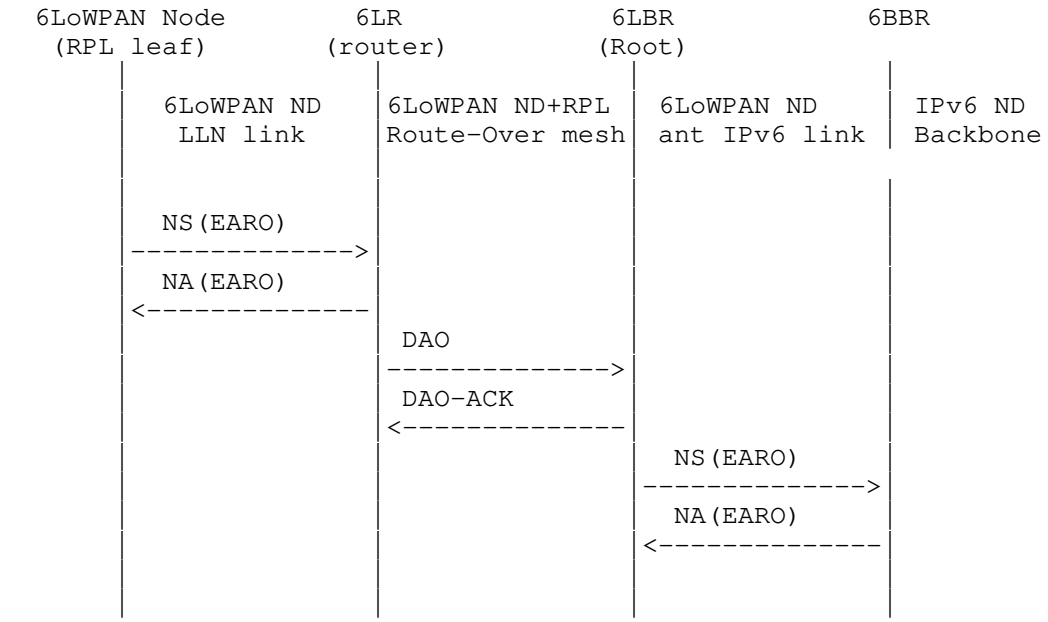

Figure 7: Next Registration Flow over Multi-Link Subnet

As the network builds up, a node should start as a leaf to join the RPL network, and may later turn into both a RPL-capable router and a 6LR, so as to accept leaf nodes to recursively join the network.

4.3. TSCH and 6top

4.3.1. 6top

6TiSCH expects a high degree of scalability together with a distributed routing functionality based on RPL. To achieve this goal, the spectrum must be allocated in a way that allows for spatial reuse between zones that will not interfere with one another. In a large and spatially distributed network, a 6TiSCH node is often in a good position to determine usage of the spectrum in its vicinity.

With 6TiSCH, the abstraction of an IPv6 link is implemented as a pair of bundles of cells, one in each direction. IP Links are only enabled between RPL parents and children. The 6TiSCH operation is

optimal when the size of a bundle is such that both the energy wasted in idle listening and the packet drops due to congestion loss are minimized, while packets are forwarded within an acceptable latency.

Use cases for distributed routing are often associated with a statistical distribution of best-effort traffic with variable needs for bandwidth on each individual link. The 6TiSCH operation can remain optimal if RPL parents can adjust dynamically, and with enough reactivity to match the variations of best-effort traffic, the amount of bandwidth that is used to communicate between themselves and their children, in both directions. In turn, the agility to fulfill the needs for additional cells improves when the number of interactions with other devices and the protocol latencies are minimized.

6top is a logical link control sitting between the IP layer and the TSCH MAC layer, which provides the link abstraction that is required for IP operations. The 6top protocol, 6P, which is specified in [RFC8480], is one of the services provided by 6top. In particular, the 6top services are available over a management API that enables an external management entity to schedule cells and slotframes, and allows the addition of complementary functionality, for instance a Scheduling Function that manages a dynamic schedule management based on observed resource usage as discussed in Section 4.4.2. For this purpose, the 6TiSCH architecture differentiates "soft" cells and "hard" cells.

4.3.1.1. Hard Cells

"Hard" cells are cells that are owned and managed by a separate scheduling entity (e.g., a PCE) that specifies the slotOffset/ channelOffset of the cells to be added/moved/deleted, in which case 6top can only act as instructed, and may not move hard cells in the TSCH schedule on its own.

4.3.1.2. Soft Cells

In contrast, "soft" cells are cells that 6top can manage locally. 6top contains a monitoring process which monitors the performance of cells, and can add, remove soft cells in the TSCH schedule to adapt to the traffic needs, or move one when it performs poorly. To reserve a soft cell, the higher layer does not indicate the exact slotOffset/channelOffset of the cell to add, but rather the resulting bandwidth and QoS requirements. When the monitoring process triggers a cell reallocation, the two neighbor devices communicating over this cell negotiate its new position in the TSCH schedule.

4.3.2. Scheduling Functions and the 6top protocol

In the case of soft cells, the cell management entity that controls the dynamic attribution of cells to adapt to the dynamics of variable rate flows is called a Scheduling Function (SF).

There may be multiple SFs with more or less aggressive reaction to the dynamics of the network.

An SF may be seen as divided between an upper bandwidth adaptation logic that is not aware of the particular technology that is used to obtain and release bandwidth, and an underlying service that maps those needs in the actual technology, which means mapping the bandwidth onto cells in the case of TSCH using the 6top protocol as illustrated in Figure 8.

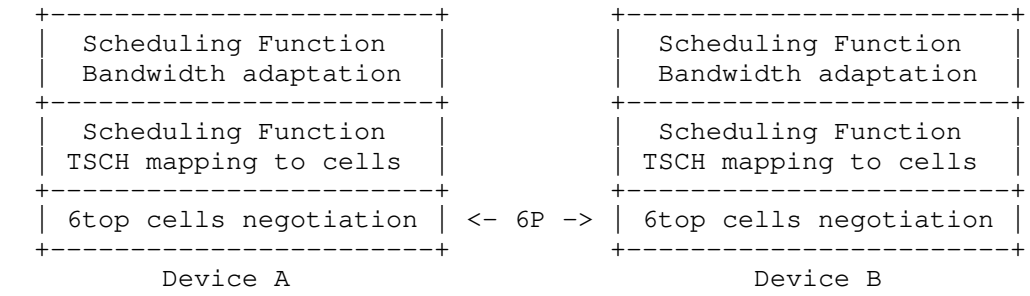

Figure 8: SF/6P stack in 6top

The SF relies on 6top services that implement the 6top Protocol (6P) [RFC8480] to negotiate the precise cells that will be allocated or freed based on the schedule of the peer. It may be for instance that a peer wants to use a particular time slot that is free in its schedule, but that timeslot is already in use by the other peer for a communication with a third party on a different cell. 6P enables the peers to find an agreement in a transactional manner that ensures the final consistency of the nodes state.

[MSF] is one of the possible scheduling functions. MSF uses the rendez-vous slot from [RFC8180] for network discovery, neighbor discovery, and any other broadcast.

For basic unicast communication with any neighbor, each node uses a receive cell at a well-known slotOffset/channelOffset, derived from a hash of their own MAC address. Nodes can reach any neighbor by installing a transmit (shared) cell with slotOffset/channelOffset derived from the neighbor's MAC address.

For child-parent links, MSF continuously monitors the load to/from parents and children. It then uses 6P to install/remove unicast cells whenever the current schedule appears to be under-/over-provisioned.

4.3.3. 6top and RPL Objective Function operations

An implementation of a RPL [RFC6550] Objective Function (OF), such as the RPL Objective Function Zero (OF0) [RFC6552] that is used in the Minimal 6TiSCH Configuration [RFC8180] to support RPL over a static schedule, may leverage, for its internal computation, the information maintained by 6top.

An OF may require metrics about reachability, such as the Expected Transmission Count (ETX) metric [RFC6551]. 6top creates and maintains an abstract neighbor table, and this state may be leveraged to feed an OF and/or store OF information as well. A neighbor table entry may contain a set of statistics with respect to that specific neighbor.

The neighbor information may include the time when the last packet has been received from that neighbor, a set of cell quality metrics, e.g., received signal strength indication (RSSI) or link quality indicator (LQI), the number of packets sent to the neighbor or the number of packets received from it. This information can be made available through 6top management APIs and used for instance to compute a Rank Increment that will determine the selection of the preferred parent.

6top provides statistics about the underlying layer so the OF can be tuned to the nature of the TSCH MAC layer. 6top also enables the RPL OF to influence the MAC behavior, for instance by configuring the periodicity of IEEE Std. 802.15.4 Extended Beacons (EBs). By augmenting the EB periodicity, it is possible to change the network dynamics so as to improve the support of devices that may change their point of attachment in the 6TiSCH network.

Some RPL control messages, such as the DODAG Information Object (DIO) are ICMPv6 messages that are broadcast to all neighbor nodes. With 6TiSCH, the broadcast channel requirement is addressed by 6top by configuring TSCH to provide a broadcast channel, as opposed to, for instance, piggybacking the DIO messages in Layer-2 Enhanced Beacons (EBs), which would produce undue timer coupling among layers, packet size issues and could conflict with the policy of production networks where EBs are mostly eliminated to conserve energy.

4.3.4. Network Synchronization

Nodes in a TSCH network must be time synchronized. A node keeps synchronized to its time source neighbor through a combination of frame-based and acknowledgment-based synchronization. To maximize battery life and network throughput, it is advisable that RPL ICMP discovery and maintenance traffic (governed by the trickle timer) be somehow coordinated with the transmission of time synchronization packets (especially with enhanced beacons).

This could be achieved through an interaction of the 6top sublayer and the RPL objective Function, or could be controlled by a management entity.

Time distribution requires a loop-free structure. Nodes taken in a synchronization loop will rapidly desynchronize from the network and become isolated. 6TiSCH uses a RPL DAG with a dedicated global Instance for the purpose of time synchronization. That Instance is referred to as the Time Synchronization Global Instance (TSGI). The TSGI can be operated in either of the 3 modes that are detailed in section 3.1.3 of RPL [RFC6550], "Instances, DODAGs, and DODAG Versions". Multiple uncoordinated DODAGs with independent Roots may be used if all the Roots share a common time source such as the Global Positioning System (GPS).

In the absence of a common time source, the TSGI should form a single DODAG with a virtual Root. A backbone network is then used to synchronize and coordinate RPL operations between the backbone routers that act as sinks for the LLN. Optionally, RPL's periodic operations may be used to transport the network synchronization. This may mean that 6top would need to trigger (override) the trickle timer if no other traffic has occurred for such a time that nodes may get out of synchronization.

A node that has not joined the TSGI advertises a MAC level Join Priority of 0xFF to notify its neighbors that is not capable of serving as time parent. A node that has joined the TSGI advertises a MAC level Join Priority set to its DAGRank() in that Instance, where DAGRank() is the operation specified in section 3.5.1 of [RFC6550], "Rank Comparison".

The provisioning of a RPL Root is out of scope for both RPL and this Architecture, whereas RPL enables to propagate configuration information down the DODAG. This applies to the TSGI as well; a Root is configured or obtains by unspecified means the knowledge of the RPLInstanceID for the TSGI. The Root advertises its DagRank in the TSGI, that must be less than 0xFF, as its Join Priority in its IEEE Std. 802.15.4 Extended Beacons (EB).

A node that reads a Join Priority of less than 0xFF should join the neighbor with the lesser Join Priority and use it as time parent. If the node is configured to serve as time parent, then the node should join the TSGI, obtain a Rank in that Instance and start advertising its own DagRank in the TSGI as its Join Priority in its EBs.

4.3.5. Slotframes and CDU matrix

6TiSCH enables IPv6 best effort (stochastic) transmissions over a MAC layer that is also capable of scheduled (deterministic) transmissions. A window of time is defined around the scheduled transmission where the medium must, as much as practically feasible, be free of contending energy to ensure that the medium is free of contending packets when time comes for a scheduled transmission. One simple way to obtain such a window is to format time and frequencies in cells of transmission of equal duration. This is the method that is adopted in IEEE Std. 802.15.4 TSCH as well as the Long Term Evolution (LTE) of cellular networks.

The 6TiSCH architecture defines a global concept that is called a Channel Distribution and Usage (CDU) matrix to describe that formatting of time and frequencies,

A CDU matrix is defined centrally as part of the network definition. It is a matrix of cells with a height equal to the number of available channels (indexed by ChannelOffsets) and a width (in timeslots) that is the period of the network scheduling operation (indexed by slotOffsets) for that CDU matrix. There are different models for scheduling the usage of the cells, which place the responsibility of avoiding collisions either on a central controller or on the devices themselves, at an extra cost in terms of energy to scan for free cells (more in Section 4.4).

The size of a cell is a timeslot duration, and values of 10 to 15 milliseconds are typical in 802.15.4 TSCH to accommodate for the transmission of a frame and an ack, including the security validation on the receive side which may take up to a few milliseconds on some device architecture.

A CDU matrix iterates over and over with a well-known channel rotation called the hopping sequence. In a given network, there might be multiple CDU matrices that operate with different width, so they have different durations and represent different periodic operations. It is recommended that all CDU matrices in a 6TiSCH domain operate with the same cell duration and are aligned, so as to reduce the chances of interferences from the Slotted ALOHA operations. The knowledge of the CDU matrices is shared between all the nodes and used in particular to define slotframes.

A slotframe is a MAC-level abstraction that is common to all nodes and contains a series of timeslots of equal length and precedence. It is characterized by a slotframe_ID, and a slotframe_size. A slotframe aligns to a CDU matrix for its parameters, such as number and duration of timeslots.

Multiple slotframes can coexist in a node schedule, i.e., a node can have multiple activities scheduled in different slotframes. A slotframe is associated with a priority that may be related to the precedence of different 6TiSCH topologies. The slotframes may be aligned to different CDU matrices and thus have different width. There is typically one slotframe for scheduled traffic that has the highest precedence and one or more slotframe(s) for RPL traffic. The timeslots in the slotframe are indexed by the SlotOffset; the first cell is at SlotOffset 0.

When a packet is received from a higher layer for transmission, 6top inserts that packet in the outgoing queue which matches the packet best (Differentiated Services [RFC2474] can therefore be used). At each scheduled transmit slot, 6top looks for the frame in all the outgoing queues that best matches the cells. If a frame is found, it is given to the TSCH MAC for transmission.

4.3.6. Distributing the reservation of cells

The 6TiSCH architecture introduces the concept of chunks (Section 2.1) to distribute the allocation of the spectrum for a whole group of cells at a time. The CDU matrix is formatted into a set of chunks, possibly as illustrated in Figure 9, each of the chunks identified uniquely by a chunk-ID. The knowledge of this formatting is shared between all the nodes in a 6TiSCH network. It could be conveyed during the join process, or codified into a profile document, or obtained using some other mechanism. This is as opposed to static scheduling that refers to the pre-programmed mechanism that is specified in [RFC8180] and pre-exists to the distribution of the chunk formatting.

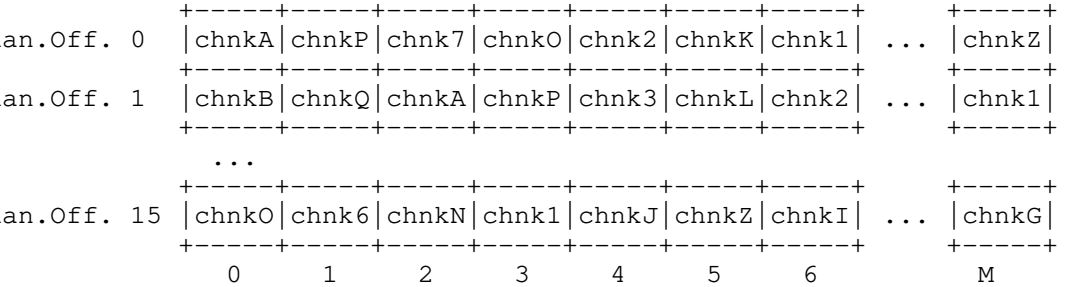

Figure 9: CDU matrix Partitioning in Chunks

The 6TiSCH Architecture envisions a protocol that enables chunk ownership appropriation whereby a RPL parent discovers a chunk that is not used in its interference domain, claims the chunk, and then defends it in case another RPL parent would attempt to appropriate it while it is in use. The chunk is the basic unit of ownership that is used in that process.

As a result of the process of chunk ownership appropriation, the RPL parent has exclusive authority to decide which cell in the appropriated chunk can be used by which node in its interference domain. In other words, it is implicitly delegated the right to manage the portion of the CDU matrix that is represented by the chunk.

Initially, those cells are added to the heap of free cells, then dynamically placed into existing bundles, in new bundles, or allocated opportunistically for one transmission.

Note that a PCE is expected to have precedence in the allocation, so that a RPL parent would only be able to obtain portions that are not in-use by the PCE.

4.4. Schedule Management Mechanisms

6TiSCH uses 4 paradigms to manage the TSCH schedule of the LLN nodes: Static Scheduling, neighbor-to-neighbor Scheduling, remote monitoring and scheduling management, and Hop-by-hop scheduling. Multiple mechanisms are defined that implement the associated Interaction Models, and can be combined and used in the same LLN. Which mechanism(s) to use depends on application requirements.

4.4.1. Static Scheduling

In the simplest instantiation of a 6TiSCH network, a common fixed schedule may be shared by all nodes in the network. Cells are shared, and nodes contend for slot access in a slotted ALOHA manner.

A static TSCH schedule can be used to bootstrap a network, as an initial phase during implementation, or as a fall-back mechanism in case of network malfunction. This schedule is pre-established, for instance decided by a network administrator based on operational needs. It can be pre-configured into the nodes, or, more commonly, learned by a node when joining the network using standard IEEE Std. 802.15.4 Information Elements (IE). Regardless, the schedule remains unchanged after the node has joined a network. RPL is used on the resulting network. This "minimal" scheduling mechanism that implements this paradigm is detailed in [RFC8180].

4.4.2. Neighbor-to-neighbor Scheduling

In the simplest instantiation of a 6TiSCH network described in Section 4.4.1, nodes may expect a packet at any cell in the schedule and will waste energy idle listening. In a more complex instantiation of a 6TiSCH network, a matching portion of the schedule is established between peers to reflect the observed amount of transmissions between those nodes. The aggregation of the cells between a node and a peer forms a bundle that the 6top layer uses to implement the abstraction of a link for IP. The bandwidth on that link is proportional to the number of cells in the bundle.

If the size of a bundle is configured to fit an average amount of bandwidth, peak traffic is dropped. If the size is configured to allow for peak emissions, energy is be wasted idle listening.

As discussed in more details in Section 4.3, the 6top Protocol [RFC8480] specifies the exchanges between neighbor nodes to reserve soft cells to transmit to one another, possibly under the control of a Scheduling Function (SF). Because this reservation is done without global knowledge of the schedule of other nodes in the LLN, scheduling collisions are possible.

And as discussed in Section 4.3.2, an optional Scheduling Function (SF) is used to monitor bandwidth usage and perform requests for dynamic allocation by the 6top sublayer. The SF component is not part of the 6top sublayer. It may be collocated on the same device or may be partially or fully offloaded to an external system. The "6TiSCH Minimal Scheduling Function (MSF)" [MSF] provides a simple scheduling function that can be used by default by devices that support dynamic scheduling of soft cells.

Monitoring and relocation is done in the 6top layer. For the upper layer, the connection between two neighbor nodes appears as a number of cells. Depending on traffic requirements, the upper layer can request 6top to add or delete a number of cells scheduled to a particular neighbor, without being responsible for choosing the exact slotOffset/channelOffset of those cells.

4.4.3. Remote Monitoring and Schedule Management

Remote monitoring and Schedule Management refers to a DetNet/SDN model whereby an NME and a scheduling entity, associated with a PCE, reside in a central controller and interact with the 6top layer to control IPv6 Links and Tracks (Section 4.5) in a 6TiSCH network. The composite centralized controller can assign physical resources (e.g., buffers and hard cells) to a particular Track to optimize the reliability within a bounded latency for a well-specified flow.

The work at the 6TiSCH WG focused on non-deterministic traffic and did not provide the generic data model that is necessary for the controller to monitor and manage resources of the 6top sublayer. This is deferred to future work, see Appendix A.1.2.

With respect to Centralized routing and scheduling, it is envisioned that the related component of the 6TiSCH Architecture would be an extension of the DetNet Architecture [RFC8655], which studies Layer-3 aspects of Deterministic Networks, and covers networks that span multiple Layer-2 domains.

The DetNet architecture is a form of Software Defined Networking (SDN) Architecture and is composed of three planes, a (User) Application Plane, a Controller Plane (where the PCE operates), and a Network Plane which can represent a 6TiSCH LLN.

Software-Defined Networking (SDN): Layers and Architecture Terminology [RFC7426] proposes a generic representation of the SDN architecture that is reproduced in Figure 10.

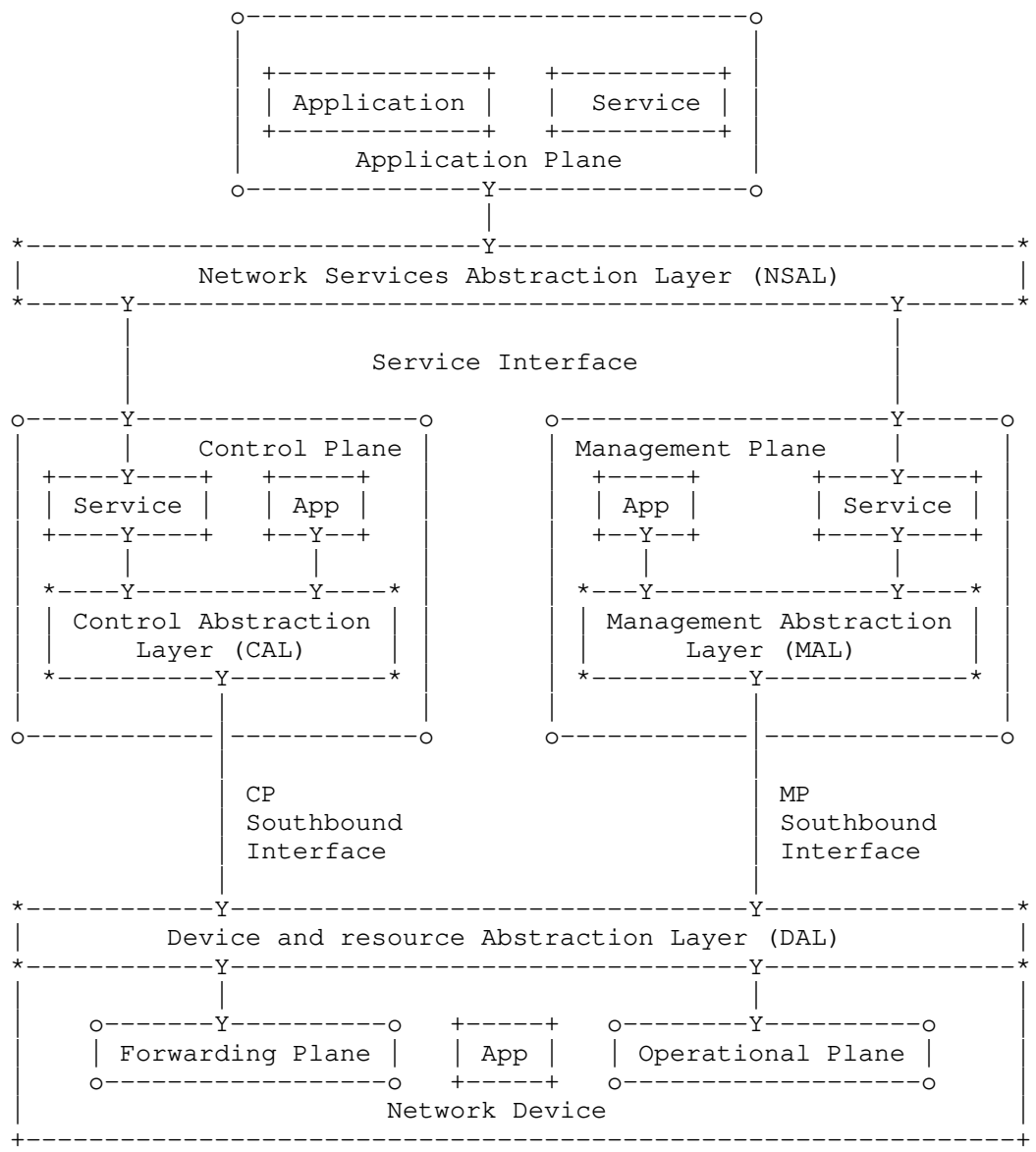

Figure 10: SDN Layers and Architecture Terminology per RFC 7426

The PCE establishes end-to-end Tracks of hard cells, which are described in more details in Section 4.6.1.

The DetNet work is expected to enable end to end Deterministic Path across heterogeneous network. This can be for instance a 6TiSCH LLN and an Ethernet Backbone.

This model fits the 6TiSCH extended configuration, whereby a 6BBR federates multiple 6TiSCH LLN in a single subnet over a backbone that can be, for instance, Ethernet or Wi-Fi. In that model, 6TiSCH 6BBRs synchronize with one another over the backbone, so as to ensure that the multiple LLNs that form the IPv6 subnet stay tightly synchronized.

If the Backbone is Deterministic, then the Backbone Router ensures that the end-to-end deterministic behavior is maintained between the LLN and the backbone. It is the responsibility of the PCE to compute a deterministic path and to end across the TSCH network and an IEEE Std. 802.1 TSN Ethernet backbone, and that of DetNet to enable end-to-end deterministic forwarding.

4.4.4. Hop-by-hop Scheduling

A node can reserve a Track (Section 4.5) to one or more destination(s) that are multiple hops away by installing soft cells at each intermediate node. This forms a Track of soft cells. A Track Scheduling Function above the 6top sublayer of each node on the Track is needed to monitor these soft cells and trigger relocation when needed.

This hop-by-hop reservation mechanism is expected to be similar in essence to [RFC3209] and/or [RFC4080]/[RFC5974]. The protocol for a node to trigger hop-by-hop scheduling is not yet defined.

4.5. On Tracks

The architecture introduces the concept of a Track, which is a directed path from a source 6TiSCH node to one or more destination 6TiSCH node(s) across a 6TiSCH LLN.

A Track is the 6TiSCH instantiation of the concept of a Deterministic Path as described in [RFC8655]. Constrained resources such as memory buffers are reserved for that Track in intermediate 6TiSCH nodes to avoid loss related to limited capacity. A 6TiSCH node along a Track not only knows which bundles of cells it should use to receive packets from a previous hop, but also knows which bundle(s) it should use to send packets to its next hop along the Track.

4.5.1. General Behavior of Tracks

A Track is associated with Layer-2 bundles of cells with related schedules and logical relationships and that ensure that a packet that is injected in a Track will progress in due time all the way to destination.

Multiple cells may be scheduled in a Track for the transmission of a single packet, in which case the normal operation of IEEE Std. 802.15.4 Automatic Repeat-reQuest (ARQ) can take place; the acknowledgment may be omitted in some cases, for instance if there is no scheduled cell for a possible retry.

There are several benefits for using a Track to forward a packet from a source node to the destination node.

1. Track forwarding, as further described in Section 4.6.1, is a Layer-2 forwarding scheme, which introduces less process delay and overhead than Layer-3 forwarding scheme. Therefore, LLN Devices can save more energy and resource, which is critical for resource constrained devices.

2. Since channel resources, i.e., bundles of cells, have been reserved for communications between 6TiSCH nodes of each hop on the Track, the throughput and the maximum latency of the traffic along a Track are guaranteed and the jitter is maintained small.

3. By knowing the scheduled time slots of incoming bundle(s) and outgoing bundle(s), 6TiSCH nodes on a Track could save more energy by staying in sleep state during in-active slots.

4. Tracks are protected from interfering with one another if a cell is scheduled to belong to at most one Track, and congestion loss is avoided if at most one packet can be presented to the MAC to use that cell. Tracks enhance the reliability of transmissions and thus further improve the energy consumption in LLN Devices by reducing the chances of retransmission.

4.5.2. Serial Track

A Serial (or simple) Track is the 6TiSCH version of a circuit; a bundle of cells that are programmed to receive (RX-cells) is uniquely paired to a bundle of cells that are set to transmit (TX-cells), representing a Layer-2 forwarding state which can be used regardless of the network layer protocol. A Serial Track is thus formed end-to-end as a succession of paired bundles, a receive bundle from the previous hop and a transmit bundle to the next hop along the Track.

For a given iteration of the device schedule, the effective channel of the cell is obtained by following in a loop a well-known hopping sequence that started at Epoch time at the channelOffset of the cell, which results in a rotation of the frequency that used for transmission. The bundles may be computed so as to accommodate both variable rates and retransmissions, so they might not be fully used in the iteration of the schedule.

4.5.3. Complex Track with Replication and Elimination

The art of Deterministic Networks already include Packet Replication and Elimination techniques. Example standards include the Parallel Redundancy Protocol (PRP) and the High-availability Seamless Redundancy (HSR) [IEC62439]. Similarly, and as opposed to a Serial Track that is a sequence of nodes and links, a Complex Track is shaped as a directed acyclic graph towards one or more destination(s) to support multi-path forwarding and route around failures.

A Complex Track may branch off over non congruent branches for the purpose of multicasting, and/or redundancy, in which case it reconverges later down the path. This enables the Packet Replication, Elimination and Ordering Functions (PREOF) defined by Detnet. Packet ARQ, Replication, Elimination and Overhearing (PAREO) adds radio-specific capabilities of Layer-2 ARQ and promiscuous listening to redundant transmissions to compensate for the lossiness of the medium and meet industrial expectations of a Reliable and Available Wireless network. Combining PAREO and PREOF, a Track may extend beyond the 6TiSCH network in a larger DetNet network.

In the art of TSCH, a path does not necessarily support PRE but it is almost systematically multi-path. This means that a Track is scheduled so as to ensure that each hop has at least two forwarding solutions, and the forwarding decision is to try the preferred one and use the other in case of Layer-2 transmission failure as detected by ARQ. Similarly, at each 6TiSCH hop along the Track, the PCE may schedule more than one timeslot for a packet, so as to support Layer-2 retries (ARQ). It is also possible that the field device only uses the second branch if sending over the first branch fails.

4.5.4. DetNet End-to-end Path

Ultimately, DetNet should enable to extend a Track beyond the 6TiSCH LLN as illustrated in Figure 11. In that example, a Track that is laid out from a field device in a 6TiSCH network to an IoT gateway that is located on an 802.1 Time-Sensitive Networking (TSN) backbone. A 6TiSCH-Aware DetNet Service Layer handles the Packet Replication, Elimination, and Ordering Functions over the DODAG that forms a Track.

The Replication function in the 6TiSCH Node sends a copy of each packet over two different branches, and the PCE schedules each hop of both branches so that the two copies arrive in due time at the gateway. In case of a loss on one branch, hopefully the other copy of the packet still makes it in due time. If two copies make it to the IoT gateway, the Elimination function in the gateway ignores the extra packet and presents only one copy to upper layers.

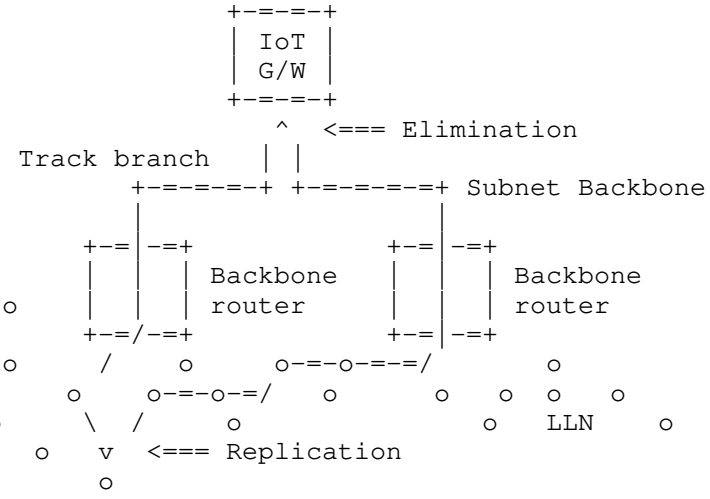

Figure 11: Example End-to-End DetNet Track

4.5.5. Cell Reuse

The 6TiSCH architecture provides means to avoid waste of cells as well as overflows in the transmit bundle of a Track, as follows:

A TX-cell that is not needed for the current iteration may be reused opportunistically on a per-hop basis for routed packets. When all of the frame that were received for a given Track are effectively transmitted, any available TX-cell for that Track can be reused for upper layer traffic for which the next-hop router matches the next hop along the Track. In that case, the cell that is being used is effectively a TX-cell from the Track, but the short address for the destination is that of the next-hop router.

It results in a frame that is received in a RX-cell of a Track with a destination MAC address set to this node as opposed to the broadcast MAC address must be extracted from the Track and delivered to the upper layer. Note that a frame with an unrecognized destination MAC address is dropped at the lower MAC layer and thus is not received at the 6top sublayer.

On the other hand, it might happen that there are not enough TX-cells in the transmit bundle to accommodate the Track traffic, for instance if more retransmissions are needed than provisioned. In that case, and if the frame transports an IPv6 packet, then it can be placed for transmission in the bundle that is used for Layer-3 traffic towards the next hop along the Track. The MAC address should be set to the next-hop MAC address to avoid confusion.

It results in a frame that is received over a Layer-3 bundle may be in fact associated to a Track. In a classical IP link such as an Ethernet, off-Track traffic is typically in excess over reservation to be routed along the non-reserved path based on its QoS setting. But with 6TiSCH, since the use of the Layer-3 bundle may be due to transmission failures, it makes sense for the receiver to recognize a frame that should be re-Tracked, and to place it back on the appropriate bundle if possible. . A frame is re-Tracked by scheduling it for transmission over the transmit bundle associated to the Track, with the destination MAC address set to broadcast.

4.6. Forwarding Models

By forwarding, this document means the per-packet operation that allows to deliver a packet to a next hop or an upper layer in this node. Forwarding is based on pre-existing state that was installed as a result of a routing computation Section 4.7. 6TiSCH supports three different forwarding model:(G-MPLS) Track Forwarding, (classical) IPv6 Forwarding and (6LoWPAN) Fragment Forwarding.

4.6.1. Track Forwarding

Forwarding along a Track can be seen as a Generalized Multi-protocol Label Switching (G-MPLS) operation in that the information used to switch a frame is not an explicit label, but rather related to other properties of the way the packet was received, a particular cell in the case of 6TiSCH. As a result, as long as the TSCH MAC (and Layer-2 security) accepts a frame, that frame can be switched regardless of the protocol, whether this is an IPv6 packet, a 6LoWPAN fragment, or a frame from an alternate protocol such as WirelessHART or ISA100.11a.

A data frame that is forwarded along a Track normally has a destination MAC address that is set to broadcast - or a multicast address depending on MAC support. This way, the MAC layer in the intermediate nodes accepts the incoming frame and 6top switches it without incurring a change in the MAC header. In the case of IEEE Std. 802.15.4, this means effectively broadcast, so that along the Track the short address for the destination of the frame is set to 0xFFFF.

There are 2 modes for a Track, an IPv6 native mode and a protocol-independant tunnel mode.

4.6.1.1. Native Mode

In native mode, the Protocol Data Unit (PDU) is associated with flow-dependent meta-data that refers uniquely to the Track, so the 6top sublayer can place the frame in the appropriate cell without ambiguity. In the case of IPv6 traffic, this flow identification may be done using a 6-tuple as discussed in [I-D.ietf-detnet-ip]. In particular, implementations of this document should support identification of DetNet flows based on the IPv6 Flow Label field.

The flow follows a Track which identification is done using a RPL Instance (see section 3.1.3 of [RFC6550]), signaled in a RPL Packet Information (more in section 11.2.2.1 of [RFC6550]) and the destination address in the case of a local instance. One or more flows may be placed in a same Track and the Track identification (TrackID + owner) may be placed in an IP-in-IP encapsulation. The forwarding operation is based on the Track and does not depend on the flow therein.

The Track identification is validated at egress before restoring the destination MAC address (DMAC) and punting to the upper layer.

Figure 12 illustrates the Track Forwarding operation which happens at the 6top sublayer, below IP.

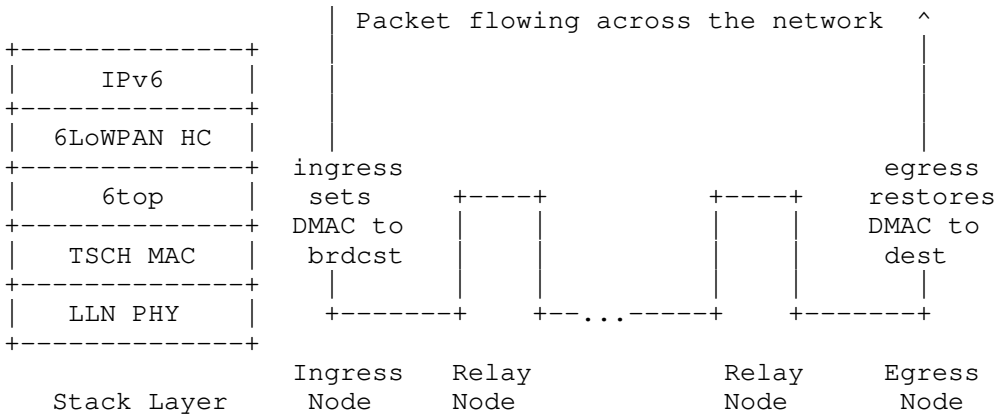

Figure 12: Track Forwarding, Native Mode

4.6.1.2. Tunnel Mode

In tunnel mode, the frames originate from an arbitrary protocol over a compatible MAC that may or may not be synchronized with the 6TiSCH network. An example of this would be a router with a dual radio that is capable of receiving and sending WirelessHART or ISA100.11a frames with the second radio, by presenting itself as an access Point or a Backbone Router, respectively. In that mode, some entity (e.g., PCE) can coordinate with a WirelessHART Network Manager or an ISA100.11a System Manager to specify the flows that are transported.

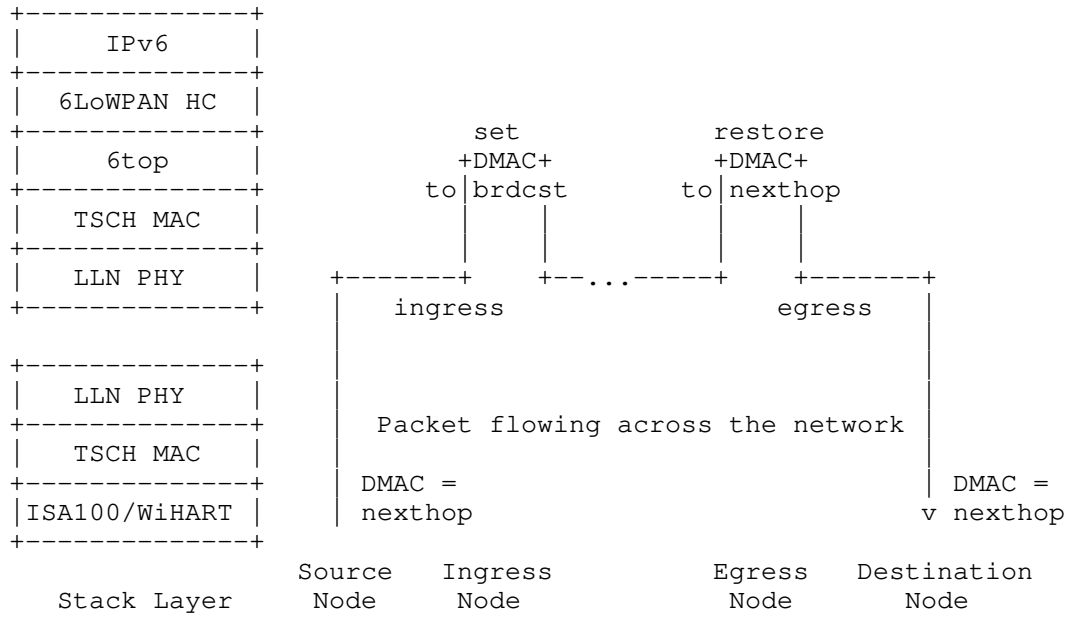

Figure 13: Track Forwarding, Tunnel Mode

In that case, the TrackID that identifies the Track at the ingress 6TiSCH router is derived from the RX-cell. The DMAC is set to this node but the TrackID indicates that the frame must be tunneled over a particular Track so the frame is not passed to the upper layer. Instead, the DMAC is forced to broadcast and the frame is passed to the 6top sublayer for switching.

At the egress 6TiSCH router, the reverse operation occurs. Based on tunneling information of the Track, which may for instance indicate that the tunneled datagram is an IP packet, the datagram is passed to the appropriate Link-Layer with the destination MAC restored.

4.6.1.3. Tunneling Information

Tunneling information coming with the Track configuration provides the destination MAC address of the egress endpoint as well as the tunnel mode and specific data depending on the mode, for instance a service access point for frame delivery at egress.

If the tunnel egress point does not have a MAC address that matches the configuration, the Track installation fails.

If the Layer-3 destination address belongs to the tunnel termination, then it is possible that the IPv6 address of the destination is compressed at the 6LoWPAN sublayer based on the MAC address. Restoring the wrong MAC address at the egress would then also result in the wrong IP address in the packet after decompression. For that reason, a packet can be injected in a Track only if the destination MAC address is effectively that of the tunnel egress point. It is thus mandatory for the ingress router to validate that the MAC address that was used at the 6LoWPAN sublayer for compression matches that of the tunnel egress point before it overwrites it to broadcast. The 6top sublayer at the tunnel egress point reverts that operation to the MAC address obtained from the tunnel information.

4.6.2. IPv6 Forwarding

As the packets are routed at Layer-3, traditional QoS and Active Queue Management (AQM) operations are expected to prioritize flows.

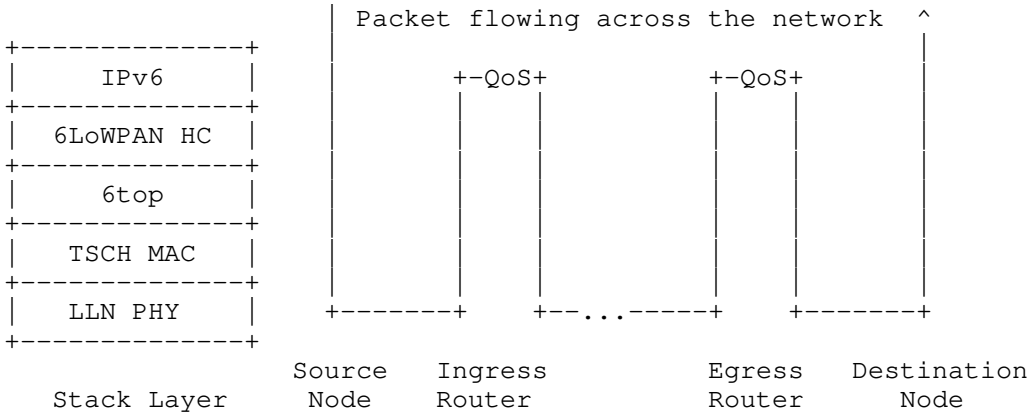

Figure 14: IP Forwarding

4.6.3. Fragment Forwarding

Considering that per section 4 of [RFC4944] 6LoWPAN packets can be as large as 1280 bytes (the IPv6 minimum MTU), and that the non-storing mode of RPL implies Source Routing that requires space for routing headers, and that a IEEE Std. 802.15.4 frame with security may carry in the order of 80 bytes of effective payload, an IPv6 packet might be fragmented into more than 16 fragments at the 6LoWPAN sublayer.

This level of fragmentation is much higher than that traditionally experienced over the Internet with IPv4 fragments, where fragmentation is already known as harmful.

In the case to a multihop route within a 6TiSCH network, Hop-by-Hop recomposition occurs at each hop to reform the packet and route it. This creates additional latency and forces intermediate nodes to store a portion of a packet for an undetermined time, thus impacting critical resources such as memory and battery.

[MIN-FRAG] describes a framework for forwarding fragments end-to-end across a 6TiSCH route-over mesh. Within that framework, [I-D.ietf-lwig-6lowpan-virtual-reassembly] details a virtual reassembly buffer mechanism whereby the datagram tag in the 6LoWPAN Fragment is used as a label for switching at the 6LoWPAN sublayer.

Building on this technique, [RECOV-FRAG] introduces a new format for 6LoWPAN fragments that enables the selective recovery of individual fragments, and allows for a degree of flow control based on an Explicit Congestion Notification.

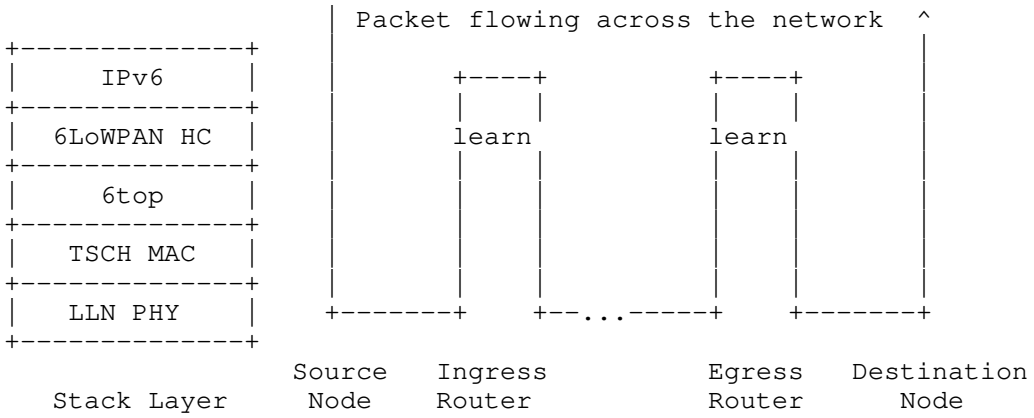

Figure 15: Forwarding First Fragment

In that model, the first fragment is routed based on the IPv6 header that is present in that fragment. The 6LoWPAN sublayer learns the next hop selection, generates a new datagram tag for transmission to the next hop, and stores that information indexed by the incoming MAC address and datagram tag. The next fragments are then switched based on that stored state.

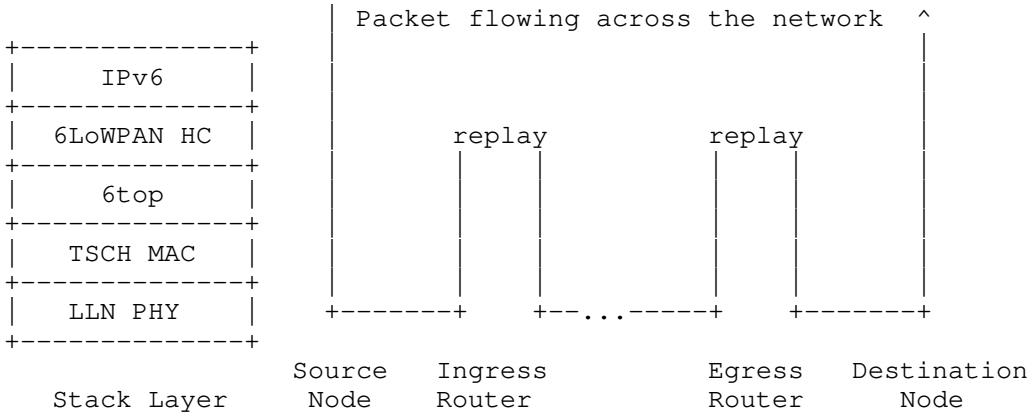

Figure 16: Forwarding Next Fragment

A bitmap and an ECN echo in the end-to-end acknowledgment enable the source to resend the missing fragments selectively. The first fragment may be resent to carve a new path in case of a path failure. The ECN echo set indicates that the number of outstanding fragments should be reduced.

4.7. Advanced 6TiSCH Routing

4.7.1. Packet Marking and Handling

All packets inside a 6TiSCH domain must carry the RPLInstanceID that identifies the 6TiSCH topology (e.g., a Track) that is to be used for routing and forwarding that packet. The location of that information must be the same for all packets forwarded inside the domain.

For packets that are routed by a PCE along a Track, the tuple formed by 1) (typically) the IPv6 source or (possibly) destination address in the IPv6 Header and 2) a local RPLInstanceID in the RPI that serves as TrackID, identify uniquely the Track and associated transmit bundle.

For packets that are routed by RPL, that information is the RPLInstanceID which is carried in the RPL Packet Information (RPI), as discussed in section 11.2 of [RFC6550], "Loop Avoidance and Detection". The RPI is transported by a RPL option in the IPv6 Hop-By-Hop Header [RFC6553].

A compression mechanism for the RPL packet artifacts that integrates the compression of IP-in-IP encapsulation and the Routing Header type 3 [RFC6554] with that of the RPI in a 6LoWPAN dispatch/header type is specified in [RFC8025] and [RFC8138].

Either way, the method and format used for encoding the RPLInstanceID is generalized to all 6TiSCH topological Instances, which include both RPL Instances and Tracks.

4.7.2. Replication, Retries and Elimination

6TiSCH supports the PREOF operations of elimination and reordering of packets along a complex Track, but has no requirement about whether a sequence number is tagged in the packet for that purpose. With 6TiSCH, the schedule can tell when multiple receive timeslots correspond to copies of a same packet, in which case the receiver may avoid listening to the extra copies once it had received one instance of the packet.

The semantics of the configuration will enable correlated timeslots to be grouped for transmit (and respectively receive) with a 'OR' relations, and then a 'AND' relation would be configurable between groups. The semantics is that if the transmit (and respectively receive) operation succeeded in one timeslot in a 'OR' group, then all the other timeslots in the group are ignored. Now, if there are at least two groups, the 'AND' relation between the groups indicates that one operation must succeed in each of the groups.

On the transmit side, timeslots provisioned for retries along a same branch of a Track are placed a same 'OR' group. The 'OR' relation indicates that if a transmission is acknowledged, then retransmissions of that packet should not be attempted for remaining timeslots in that group. There are as many 'OR' groups as there are branches of the Track departing from this node. Different 'OR' groups are programmed for the purpose of replication, each group corresponding to one branch of the Track. The 'AND' relation between the groups indicates that transmission over any of branches must be attempted regardless of whether a transmission succeeded in another branch. It is also possible to place cells to different next-hop routers in a same 'OR' group. This allows to route along multi-path Tracks, trying one next-hop and then another only if sending to the first fails.

On the receive side, all timeslots are programmed in a same 'OR' group. Retries of a same copy as well as converging branches for elimination are converged, meaning that the first successful reception is enough and that all the other timeslots can be ignored. A 'AND' group denotes different packets that must all be received and transmitted over the associated transmit groups within their respected 'AND' or 'OR' rules.

As an example say that we have a simple network as represented in Figure 17, and we want to enable PREOF between an ingress node I and an egress node E.

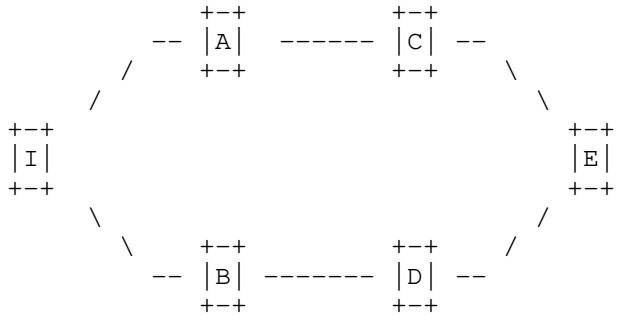

Figure 17: Scheduling PREOF on a Simple Network

The assumption for this particular problem is that a 6TiSCH node has a single radio, so it cannot perform 2 receive and/or transmit operations at the same time, even on 2 different channels.

Say we have 6 possible channels, and at least 10 timeslots per slotframe. Figure 18 shows a possible schedule whereby each transmission is retried 2 or 3 times, and redundant copies are

forwarded in parallel via A and C on the one hand, and B and D on the other, providing time diversity, spatial diversity though different physical paths, and frequency diversity.

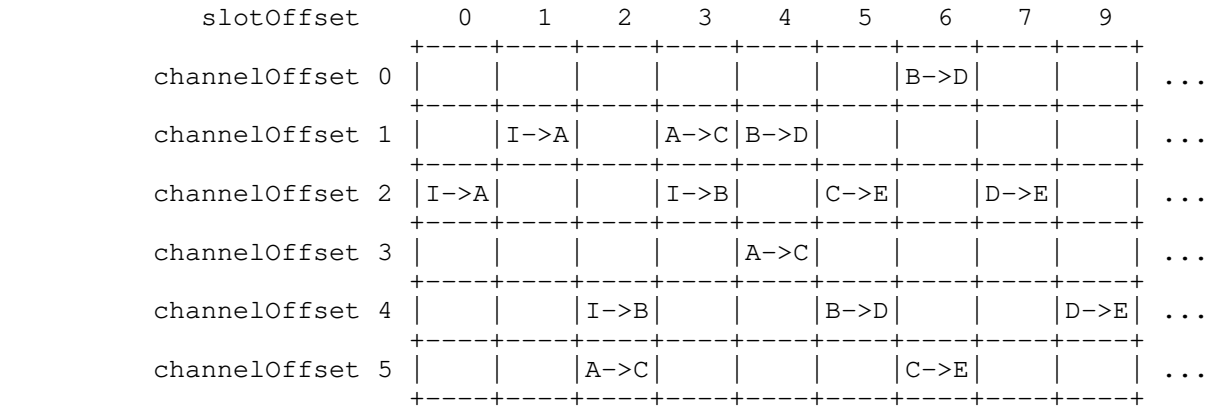

Figure 18: Example Global Schedule

This translates in a different slotframe for every node that provides the waking and sleeping times, and the channelOffset to be used when awake. Figure 19 shows the corresponding slotframe for node A.

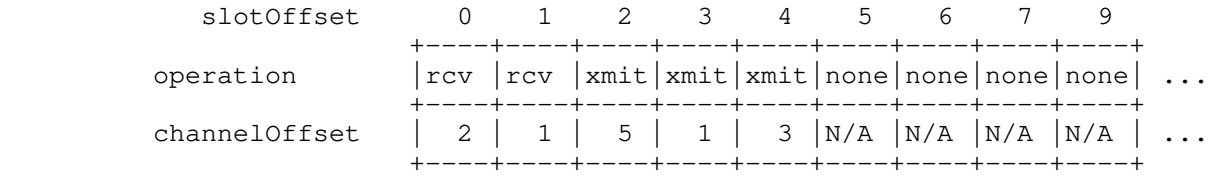

Figure 19: Example Slotframe for Node A

The logical relationship between the timeslots is given by the following table:

Node	rcv slotOffset	xmit slotOffset
I	N/A	(0 OR 1) AND (2 OR 3)
A	(0 OR 1)	(2 OR 3 OR 4)
B	(2 OR 3)	(4 OR 5 OR 6)
C	(2 OR 3 OR 4)	(5 OR 6)
D	(4 OR 5 OR 6)	(7 OR 8)
E	(5 OR 6 OR 7 OR 8)	N/A

Figure 20

5. IANA Considerations

This document does not require IANA action.

6. Security Considerations

The "Minimal Security Framework for 6TiSCH" [MIN-SECURITY] was optimized for Low-Power and TSCH operations. The reader is encouraged to review the Security Considerations section of that document, which discusses 6TiSCH security issues in more details.

6.1. Availability of Remote Services

The operation of 6TiSCH Tracks inherits its high level operation from DetNet and is subject to the observations in section 5 of [RFC8655]. The installation and the maintenance of the 6TiSCH Tracks depends on the availability of a controller with a PCE to compute and push them in the network. When that connectivity is lost, existing Tracks may continue to operate until the end of their lifetime, but cannot be removed or updated, and new Tracks cannot be installed.

In a LLN, the communication with a remote PCE may be slow and unreactive to rapid changes in the condition of the wireless communication. An attacker may introduce extra delay by selectively jamming some packets or some flows. The expectation is that the 6TiSCH Tracks enable enough redundancy to maintain the critical traffic in operation while new routes are calculated and programmed into the network.

As with DetNet in general, the communication with the PCE must be secured and should be protected against DoS attacks, including delay injection and blackholing attacks, and secured as discussed in the security considerations defined for Abstraction and Control of Traffic Engineered Networks (ACTN) in Section 9 of [RFC8453], which applies equally to DetNet and 6TiSCH. In a similar manner, the communication with the JRC must be secured and should be protected against DoS attacks when possible.

6.2. Selective Jamming

The Hopping Sequence of a TSCH network is well-known, meaning that if a rogue manages to identify a cell of a particular flow, then it may to selectively jam that cell, without impacting any other traffic. This attack can be performed at the PHY layer without any knowledge of the Layer-2 keys, and is very hard to detect and diagnose because only one flow is impacted.

[I-D.tiloca-6tisch-robust-scheduling] proposes a method to obfuscate the hopping sequence and make it harder to perpetrate that particular attack.

6.3. MAC-Layer Security

This architecture operates on IEEE Std. 802.15.4 and expects the Link-Layer security to be enabled at all times between connected devices, except for the very first step of the device join process, where a joining device may need some initial, unsecured exchanges so as to obtain its initial key material. In a typical deployment, all joined nodes use the same keys and rekeying needs to be global.

The 6TISCH Architecture relies on the join process to deny authorization of invalid nodes and preserve the integrity of the network keys. A rogue that managed to access the network can perform a large variety of attacks from DoS to injecting forged packets and routing information. "Zero-trust" properties would be highly desirable but are mostly not available at the time of this writing. [AP-ND] is a notable exception that protects the ownership of IPv6 addresses and prevents a rogue node with L2 access from stealing and injecting traffic on behalf of a legitimate node.

6.4. Time Synchronization

Time Synchronization in TSCH induces another event horizon whereby a node will only communicate with another node if they are synchronized within a guard time. The pledge discovers the synchronization of the network based on the time of reception of the beacon. If an attacker synchronizes a pledge outside of the guard time of the legitimate nodes then the pledge will never see a legitimate beacon and may not discover the attack.

As discussed in [RFC8655], measures must be taken to protect the time synchronization, and for 6TiSCH this includes ensuring that the Absolute Slot Number (ASN), which is the node’s sense of time, is not compromised. Once installed and as long as the node is synchronized to the network, ASN is implicit in the transmissions.

IEEE Std. 802.15.4 [IEEE802154] specifies that in a TSCH network, the nonce that is used for the computation of the Message Integrity Code (MIC) to secure Link-Layer frames is composed of the address of the source of the frame and of the ASN. The standard assumes that the ASN is distributed securely by other means. The ASN is not passed explicitly in the data frames and does not constitute a complete anti-replay protection. It results that upper layer protocols must provide a way to detect duplicates and cope with them.

If the receiver and the sender have a different sense of ASN, the MIC will not validate and the frame will be dropped. In that sense, TSCH induces an event horizon whereby only nodes that have a common sense of ASN can talk to one another in an authenticated manner. With 6TiSCH, the pledge discovers a tentative ASN in beacons from nodes that have already joined the network. But even if the beacon can be authenticated, the ASN cannot be trusted as it could be a replay by an attacker and thus could announce an ASN that represents a time in the past. If the pledge uses an ASN that is learned from a replayed beacon for an encrypted transmission, a nonce-reuse attack becomes possible and the network keys may be compromised.

6.5. Validating ASN

After obtaining the tentative ASN, a pledge that wishes to join the 6TiSCH network must use a join protocol to obtain its security keys. The join protocol used in 6TiSCH is the Constrained Join Protocol (CoJP). In the minimal setting defined in [MIN-SECURITY], the authentication requires a pre-shared key, based on which a secure session is derived. The CoJP exchange may also be preceded with a zero-touch handshake [I-D.ietf-6tisch-dtsecurity-zerotouch-join] in order to enable pledge joining based on certificates and/or inter-domain communication.

As detailed in Section 4.2.1, a Join Proxy (JP) helps the pledge for the join procedure by relaying the link-scope Join Request over the IP network to a Join Registrar/Coordinator (JRC) that can authenticate the pledge and validate that it is attached to the appropriate network. As a result of the CoJP exchange, the pledge is in possession of a Link-Layer material including keys and a short address, and if the ASN is known to be correct, all traffic can now be secured using CCM* [CCMstar] at the Link-Layer.

The authentication steps must be such that they cannot be replayed by an attacker, and they must not depend on the tentative ASN being valid. During the authentication, the keying material that the pledge obtains from the JRC does not provide protection against spoofed ASN. Once the pledge has obtained the keys to use in the network, it may still need to verify the ASN. If the nonce used in the Layer-2 security derives from the extended (MAC-64) address, then replaying the ASN alone cannot enable a nonce-reuse attack unless the same node is lost its state with a previous ASN. But if the nonce derives from the short address (e.g., assigned by the JRC) then the JRC must ensure that it never assigns short addresses that were already given to this or other nodes with the same keys. In other words, the network must be rekeyed before the JRC runs out of short addresses.

6.6. Network Keying and Rekeying

Section 4.2.1 provides an overview of the CoJP process described in [MIN-SECURITY] by which an LLN can be assembled in the field, having been provisioned in a lab.
[I-D.ietf-6tisch-dtsecurity-zerotouch-join] is future work that preceeds and then leverages the CoJP protocol using the [I-D.ietf-anima-constrained-voucher] constrained profile of [I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI). This later work requires a yet-to-be standardized Lighweight Authenticated Key Exchange protocol.

The CoJP protocol results in distribution of a network-wide key that is to be used with [IEEE802154] security. The details of use are described in [MIN-SECURITY] sections 9.2 and 9.3.2.

The BRSKI mechanism may lead to the use of the CoJP protocol, in which case it also results in distribution of a network-wide key. Alternatively the BRSKI mechanism may be followed by use of [I-D.ietf-ace-coap-est] to enroll certificates for each device. In that case, the certificates may be used with an [IEEE802154] key agreement protocol. The description of this mechanism, while conceptually straight forward still has significant standardization hurdles to pass.

[MIN-SECURITY] section 9.2 describes a mechanism to change (rekey) the network. There are a number of reasons to initiate a network rekey: to remove unwanted (corrupt/malicious) nodes, to recover unused 2-byte short addresses, or due to limits in encryption algorithms. For all of the mechanisms that distribute a network-wide key, rekeying is also needed on a periodic basis. In more details:

* The mechanism described in [MIN-SECURITY] section 9.2 requires advance communication between the JRC and every one of the nodes before the key change. Given that many nodes may be sleepy, this operation may take a significant amount of time, and may consume a significant portion of the available bandwidth. As such, network-wide rekeys in order to exclude nodes that have become malicious will not be particularly quick. If a rekey is already in progress, but the unwanted node has not yet been updated, then it is possible to to just continue the operation. If the unwanted node has already received the update, then the rekey operation will need to be restarted.

* The cryptographic mechanisms used by IEEE Std. 802.15.4 include the 2-byte short address in the calculation of the context. A nonce-reuse attack may become feasible if a short address is

reassigned to another node while the same network-wide keys are in operation. A network that gains and loses nodes on a regular basis is likely to reach the 65536 limit of the 2-byte (16-bit) short addresses, even if the network has only a few thousand nodes. Network planners should consider the need to rekey the network on a periodic basis in order to recover 2-byte addresses. The rekey can update the short addresses for active nodes if desired, but there is actually no need to do this as long as the key has been changed.

* With TSCH as it stands at the time of this writing, the ASN will wrap after 2^40 timeslot durations, which means with the default values around 350 years. Wrapping ASN is not expected to happen within the lifetime of most LLNs. Yet, should the ASN wrap, the network must be rekeyed to avoid a nonce-reuse attack.

* Many cipher algorithms have some suggested limits on how many bytes should be encrypted with that algorithm before a new key is used. These numbers are typically in the many to hundreds of gigabytes of data. On very fast backbone networks this becomes an important concern. On LLNs with typical data rates in the kilobits/second, this concern is significantly less. With IEEE Std. 802.15.4 as it stands at the time of this writing, the ASN will wrap before the limits of the current L2 crypto (AES-CCM-128) are reached, so the problem should never occur.

* In any fashion, if the LLN is expected to operate continuously for decades then the operators are advised to plan for the need to rekey.

Except for urgent rekeys caused by malicious nodes, the rekey operation described in [MIN-SECURITY] can be done as a background task and can be done incrementally. It is a make-before-break mechanism. The switch over to the new key is not signaled by time, but rather by observation that the new key is in use. As such, the update can take as long as needed, or occur in as short a time as practical.

7. Acknowledgments

7.1. Contributors

The co-authors of this document are listed below:

Thomas Watteyne for his contribution to the whole design, in particular on TSCH and security, and to the open source community with openWSN that he created.

Xavier Vilajosana who lead the design of the minimal support with RPL and contributed deeply to the 6top design and the G-MPLS operation of Track switching;

Kris Pister for creating TSCH and his continuing guidance through the elaboration of this design;

Malisa Vucinic for the work on the one-touch join process and his contribution to the Security Design Team;

Michael Richardson for his leadership role in the Security Design Team and his contribution throughout this document;

Tero Kivinen for his contribution to the security work in general and the security section in particular.

Maria Rita Palattella for managing the Terminology document merged into this through the work of 6TiSCH;

Simon Duquennoy for his contribution to the open source community with the 6TiSCH implementaton of contiki, and for his contribution to MSF and autonomous unicast cells.

Qin Wang who lead the design of the 6top sublayer and contributed related text that was moved and/or adapted in this document;

Rene Struik for the security section and his contribution to the Security Design Team;

Robert Assimiti for his breakthrough work on RPL over TSCH and initial text and guidance;

7.2. Special Thanks

Special thanks to Jonathan Simon, Giuseppe Piro, Subir Das and Yoshihiro Ohba for their deep contribution to the initial security work, to Yasuyuki Tanaka for his work on implementation and simulation that tremendously helped build a robust system, to Diego Dujovne for starting and leading the SF0 effort and to Tengfei Chang for evolving it in the MSF.

Special thanks also to Pat Kinney, Charlie Perkins and Bob Heile for their support in maintaining the connection active and the design in line with work happening at IEEE 802.15.

Special thanks to Ted Lemon who was the INT Area A-D while this document was initiated for his great support and help throughout, and to Suresh Krishnan who took over with that kind efficiency of his till publication.

Also special thanks to Ralph Droms who performed the first INT Area Directorate review, that was very deep and thorough and radically changed the orientations of this document, and then to Eliot Lear and Carlos Pignataro who help finalize this document in preparation to the IESG reviews, and to Gorry Fairhurst, David Mandelberg, Qin Wu, Francis Dupont, Eric Vyncke, Mirja Kuhlewind, Roman Danyliw, Benjamin Kaduk and Andrew Malis, who contributed to the final shaping of this document through the IESG review procedure.

7.3. And Do not Forget

This document is the result of multiple interactions, in particular during the 6TiSCH (bi)Weekly Interim call, relayed through the 6TiSCH mailing list at the IETF, over the course of more than 5 years.

The authors wish to thank in arbitrary order: Alaeddine Weslati, Chonggang Wang, Georgios Exarchakos, Zhuo Chen, Georgios Papadopoulos, Eric Levy-Abegnoli, Alfredo Grieco, Bert Greevenbosch, Cedric Adjih, Deji Chen, Martin Turon, Dominique Barthel, Elvis Vogli, Geraldine Texier, Guillaume Gaillard, Herman Storey, Kazushi Muraoka, Ken Bannister, Kuor Hsin Chang, Laurent Toutain, Maik Seewald, Michael Behringer, Nancy Cam Winget, Nicola Accettura, Nicolas Montavont, Oleg Hahm, Patrick Wetterwald, Paul Duffy, Peter van der Stock, Rahul Sen, Pieter de Mil, Pouria Zand, Rouhollah Nabati, Rafa Marin-Lopez, Raghuram Sudhaakar, Sedat Gormus, Shitanshu Shah, Steve Simlo, Tina Tsou, Tom Phinney, Xavier Lagrange, Ines Robles and Samita Chakrabarti for their participation and various contributions.

8. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012, <https://www.rfc-editor.org/info/rfc6551>.

[RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <https://www.rfc-editor.org/info/rfc6552>.

[RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <https://www.rfc-editor.org/info/rfc6553>.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <https://www.rfc-editor.org/info/rfc6554>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <https://www.rfc-editor.org/info/rfc6775>.

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

[RFC8025] Thubert, P., Ed. and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch", RFC 8025, DOI 10.17487/RFC8025, November 2016, <https://www.rfc-editor.org/info/rfc8025>.

[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>.

[RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <https://www.rfc-editor.org/info/rfc8138>.

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8480] Wang, Q., Ed., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)", RFC 8480, DOI 10.17487/RFC8480, November 2018, <https://www.rfc-editor.org/info/rfc8480>.

[RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <https://www.rfc-editor.org/info/rfc8453>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

[RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January 2014, <https://www.rfc-editor.org/info/rfc7102>.

[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>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC5889] Baccelli, E., Ed. and M. Townsley, Ed., "IP Addressing Model in Ad Hoc Networks", RFC 5889, DOI 10.17487/RFC5889, September 2010, <https://www.rfc-editor.org/info/rfc5889>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

[MIN-SECURITY] Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", Work in Progress, Internet-Draft, draft-ietf-6tisch-minimal-security-15, 10 December 2019, <https://tools.ietf.org/html/draft-ietf-6tisch-minimal-security-15>.

[6BBR-DRAFT] Thubert, P., Perkins, C., and E. Levy-Abegnoli, "IPv6 Backbone Router", Work in Progress, Internet-Draft, draft-ietf-6lo-backbone-router-20, 23 March 2020, <https://tools.ietf.org/html/draft-ietf-6lo-backbone-router-20>.

[RECOV-FRAG]
Thubert, P., "6LoWPAN Selective Fragment Recovery", Work in Progress, Internet-Draft, draft-ietf-6lo-fragment-recovery-21, 23 March 2020, <https://tools.ietf.org/html/draft-ietf-6lo-fragment-recovery-21>.

[MIN-FRAG] Watteyne, T., Thubert, P., and C. Bormann, "On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network", Work in Progress, Internet-Draft, draft-ietf-6lo-minimal-fragment-15, 23 March 2020, <https://tools.ietf.org/html/draft-ietf-6lo-minimal-fragment-15>.

[AP-ND] Thubert, P., Sarikaya, B., Sethi, M., and R. Struik, "Address Protected Neighbor Discovery for Low-power and Lossy Networks", Work in Progress, Internet-Draft, draft-ietf-6lo-ap-nd-23, 30 April 2020, <https://tools.ietf.org/html/draft-ietf-6lo-ap-nd-23>.

[USEofRPLinfo]
Robles, I., Richardson, M., and P. Thubert, "Using RPI Option Type, Routing Header for Source Routes and IPv6-in-IPv6 encapsulation in the RPL Data Plane", Work in Progress, Internet-Draft, draft-ietf-roll-useofrplinfo-42, 12 November 2020, <https://tools.ietf.org/html/draft-ietf-roll-useofrplinfo-42>.

[RUL-DRAFT]
Thubert, P. and M. Richardson, "Routing for RPL Leaves", Work in Progress, Internet-Draft, draft-ietf-roll-unaware-leaves-23, 10 November 2020, <https://tools.ietf.org/html/draft-ietf-roll-unaware-leaves-23>.

[ENH-BEACON]
Dujovne, D. and M. Richardson, "IEEE 802.15.4 Information Element encapsulation of 6TiSCH Join and Enrollment Information", Work in Progress, Internet-Draft, draft-ietf-6tisch-enrollment-enhanced-beacon-14, 21 February 2020, <https://tools.ietf.org/html/draft-ietf-6tisch-enrollment-enhanced-beacon-14>.

[MSF] Chang, T., Vucinic, M., Vilajosana, X., Duquennoy, S., and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)", Work in Progress, Internet-Draft, draft-ietf-6tisch-msf-18, 12 September 2020, <https://tools.ietf.org/html/draft-ietf-6tisch-msf-18>.

9. Informative References

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <https://www.rfc-editor.org/info/rfc6275>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2545] Marques, P. and F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing", RFC 2545, DOI 10.17487/RFC2545, March 1999, <https://www.rfc-editor.org/info/rfc2545>.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, DOI 10.17487/RFC3963, January 2005, <https://www.rfc-editor.org/info/rfc3963>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC4080] Hancock, R., Karagiannis, G., Loughney, J., and S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, DOI 10.17487/RFC4080, June 2005, <https://www.rfc-editor.org/info/rfc4080>.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, DOI 10.17487/RFC4919, August 2007, <https://www.rfc-editor.org/info/rfc4919>.

[RFC4903] Thaler, D., "Multi-Link Subnet Issues", RFC 4903, DOI 10.17487/RFC4903, June 2007, <https://www.rfc-editor.org/info/rfc4903>.

[RFC5974] Manner, J., Karagiannis, G., and A. McDonald, "NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling", RFC 5974, DOI 10.17487/RFC5974, October 2010, <https://www.rfc-editor.org/info/rfc5974>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing", RFC 6606, DOI 10.17487/RFC6606, May 2012, <https://www.rfc-editor.org/info/rfc6606>.

[I-D.ietf-roll-rpl-industrial-applicability] Phinney, T., Thubert, P., and R. Assimiti, "RPL applicability in industrial networks", Work in Progress, Internet-Draft, draft-ietf-roll-rpl-industrial-applicability-02, 21 October 2013, <https://tools.ietf.org/html/draft-ietf-roll-rpl-industrial-applicability-02>.

[I-D.ietf-6tisch-dtsecurity-zerotouch-join] Richardson, M., "6tisch Zero-Touch Secure Join protocol", Work in Progress, Internet-Draft, draft-ietf-6tisch-dtsecurity-zerotouch-join-04, 8 July 2019, <https://tools.ietf.org/html/draft-ietf-6tisch-dtsecurity-zerotouch-join-04>.

[I-D.ietf-core-object-security] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments

(OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-object-security-16, 6 March 2019, <https://tools.ietf.org/html/draft-ietf-core-object-security-16>.

[I-D.ietf-manet-aodvv2]
Perkins, C., Ratliff, S., Dowdell, J., Steenbrink, L., and V. Mercieca, "Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing", Work in Progress, Internet-Draft, draft-ietf-manet-aodvv2-16, 4 May 2016, <https://tools.ietf.org/html/draft-ietf-manet-aodvv2-16>.

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

[I-D.ietf-detnet-ip]
Varga, B., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "DetNet Data Plane: IP", Work in Progress, Internet-Draft, draft-ietf-detnet-ip-07, 3 July 2020, <https://tools.ietf.org/html/draft-ietf-detnet-ip-07>.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-bootstrapping-keyinfra-45, 11 November 2020, <https://tools.ietf.org/html/draft-ietf-anima-bootstrapping-keyinfra-45>.

[I-D.ietf-roll-aodv-rpl]
Anamalamudi, S., Zhang, M., Perkins, C., Anand, S., and B. Liu, "AODV based RPL Extensions for Supporting Asymmetric P2P Links in Low-Power and Lossy Networks", Work in Progress, Internet-Draft, draft-ietf-roll-aodv-rpl-08, 7 May 2020, <https://tools.ietf.org/html/draft-ietf-roll-aodv-rpl-08>.

[I-D.ietf-lwig-6lowpan-virtual-reassembly]
Bormann, C. and T. Watteyne, "Virtual reassembly buffers in 6LoWPAN", Work in Progress, Internet-Draft, draft-ietf-lwig-6lowpan-virtual-reassembly-02, 9 March 2020, <https://tools.ietf.org/html/draft-ietf-lwig-6lowpan-virtual-reassembly-02>.

[I-D.ietf-roll-dao-projection]
Thubert, P., Jadhav, R., and M. Gillmore, "Root initiated routing state in RPL", Work in Progress, Internet-Draft,

draft-ietf-roll-dao-projection-14, 2 October 2020, <https://tools.ietf.org/html/draft-ietf-roll-dao-projection-14>.

[I-D.rahul-roll-mop-ext]
Jadhav, R. and P. Thubert, "RPL Mode of Operation extension", Work in Progress, Internet-Draft, draft-rahul-roll-mop-ext-01, 9 June 2019, <https://tools.ietf.org/html/draft-rahul-roll-mop-ext-01>.

[I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-selander-ace-cose-ecdhe-14, 11 September 2019, <https://tools.ietf.org/html/draft-selander-ace-cose-ecdhe-14>.

[I-D.thubert-roll-bier]
Thubert, P., "RPL-BIER", Work in Progress, Internet-Draft, draft-thubert-roll-bier-02, 24 July 2018, <https://tools.ietf.org/html/draft-thubert-roll-bier-02>.

[I-D.thubert-bier-replication-elimination]
Thubert, P., Eckert, T., Brodard, Z., and H. Jiang, "BIER-TE extensions for Packet Replication and Elimination Function (PREF) and OAM", Work in Progress, Internet-Draft, draft-thubert-bier-replication-elimination-03, 3 March 2018, <https://tools.ietf.org/html/draft-thubert-bier-replication-elimination-03>.

[I-D.thubert-6lo-bier-dispatch]
Thubert, P., Brodard, Z., Jiang, H., and G. Texier, "A 6loRH for BitStrings", Work in Progress, Internet-Draft, draft-thubert-6lo-bier-dispatch-06, 28 January 2019, <https://tools.ietf.org/html/draft-thubert-6lo-bier-dispatch-06>.

[I-D.thubert-6man-unicast-lookup]
Thubert, P. and E. Levy-Abegnoli, "IPv6 Neighbor Discovery Unicast Lookup", Work in Progress, Internet-Draft, draft-thubert-6man-unicast-lookup-00, 29 July 2019, <https://tools.ietf.org/html/draft-thubert-6man-unicast-lookup-00>.

[I-D.pthubert-raw-problem-statement]
Thubert, P. and G. Papadopoulos, "Reliable and Available Wireless Problem Statement", Work in Progress, Internet-Draft, draft-pthubert-raw-problem-statement-04, 23 October 2019, <https://tools.ietf.org/html/draft-pthubert-raw-problem-statement-04>.

[I-D.tiloca-6tisch-robust-scheduling]
Tiloca, M., Duquennoy, S., and G. Dini, "Robust Scheduling against Selective Jamming in 6TiSCH Networks", Work in Progress, Internet-Draft, draft-tiloca-6tisch-robust-scheduling-02, 10 June 2019, <https://tools.ietf.org/html/draft-tiloca-6tisch-robust-scheduling-02>.

[I-D.ietf-ace-coap-est]
Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <https://tools.ietf.org/html/draft-ietf-ace-coap-est-18>.

[I-D.ietf-anima-constrained-voucher]
Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-09, 2 November 2020, <https://tools.ietf.org/html/draft-ietf-anima-constrained-voucher-09>.

[IEEE802154]
IEEE standard for Information Technology, "IEEE Std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks".

[CCMstar] Struik, R., "Formal Specification of the CCM* Mode of Operation", September 2004, <www.ieee802.org/15/pub/2004/15-04-0537-00-004b-formal-specification-ccm-star-mode-operation.doc>.

[IEEE802154e]
IEEE standard for Information Technology, "IEEE standard for Information Technology, IEEE Std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks, June 2011 as amended by IEEE Std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

[WirelessHART]
www.hartcomm.org, "Industrial Communication Networks - Wireless Communication Network and Communication Profiles - WirelessHART - IEC 62591", 2010.

[HART] www.hartcomm.org, "Highway Addressable remote Transducer, a group of specifications for industrial process and control devices administered by the HART Foundation".

[ISA100.11a]
ISA/ANSI, "Wireless Systems for Industrial Automation: Process Control and Related Applications - ISA100.11a-2011 - IEC 62734", 2011, <http://www.isa.org/Community/SP100WirelessSystemsforAutomation>.

[ISA100] ISA/ANSI, "ISA100, Wireless Systems for Automation", <https://www.isa.org/isa100/>.

[TEAS] IETF, "Traffic Engineering Architecture and Signaling", <https://dataTracker.ietf.org/doc/charter-ietf-teas/>.

[ANIMA] IETF, "Autonomic Networking Integrated Model and Approach", <https://dataTracker.ietf.org/doc/charter-ietf-anima/>.

[PCE] IETF, "Path Computation Element", <https://dataTracker.ietf.org/doc/charter-ietf-pce/>.

[CCAMP] IETF, "Common Control and Measurement Plane", <https://dataTracker.ietf.org/doc/charter-ietf-ccamp/>.

[AMI] US Department of Energy, "Advanced Metering Infrastructure and Customer Systems", 2006, <https://www.energy.gov/sites/prod/files/2016/12/f34/AMI%20Summary%20Report_09-26-16.pdf>.

[S-ALOHA] Roberts, L. G., "ALOHA Packet System With and Without Slots and Capture", doi 10.1145/1024916.1024920, April 1975, <https://dl.acm.org/citation.cfm?id=1024920>.

[IEC62439] IEC, "Industrial communication networks - High availability automation networks - Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR) - IEC62439-3", 2012, <https://webstore.iec.ch/publication/7018>.

Appendix A. Related Work In Progress

This document has been incremented as the work progressed following the evolution of the WG charter and the availability of dependent work. The intent was to publish when the WG concludes on the covered items. At the time of publishing the following specification are still in progress and may affect the evolution of the stack in a 6TiSCH-aware node.

A.1. Unchartered IETF work items

A.1.1. 6TiSCH Zerotouch security

The security model and in particular the zerotouch join process [I-D.ietf-6tisch-dtsecurity-zerotouch-join] depends on the ANIMA [ANIMA] Bootstrapping Remote Secure Key Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra] to enable zero-touch security provisionning; for highly constrained nodes, a minimal model based on pre-shared keys (PSK) is also available. As written to this day, it also depends on a number of documents in progress as CORE, and on "Ephemeral Diffie-Hellman Over COSE (EDHOC)" [I-D.selander-ace-cose-ecdhe], which is being considered for adoption at the LAKE WG.

A.1.2. 6TiSCH Track Setup

ROLL is now standardizing a reactive routing protocol based on RPL [I-D.ietf-roll-aodv-rpl] The need of a reactive routing protocol to establish on-demand constraint-optimized routes and a reservation protocol to establish Layer-3 Tracks is being discussed at 6TiSCH but not chartered for.

At the time of this writing, there is new work planned in the IETF to provide limited deterministic networking capabilities for wireless networks with a focus on forwarding behaviors to react quickly and locally to the changes as described in [I-D.pthubert-raw-problem-statement].

ROLL is also standardizing an extension to RPL to setup centrally-computed routes [I-D.ietf-roll-dao-projection]

The 6TiSCH Architecture should thus inherit from the DetNet [RFC8655] architecture and thus depends on it. The Path Computation Element (PCE) should be a core component of that architecture. An extension to RPL or to TEAS [TEAS] will be required to expose the 6TiSCH node capabilities and the network peers to the PCE, possibly in combination with [I-D.rahul-roll-mop-ext]. A protocol such as a lightweight PCEP or an adaptation of CCAMP [CCAMP] G-MPLS formats and

procedures could be used in combination to [I-D.ietf-roll-dao-projection] to install the Tracks, as computed by the PCE, to the 6TiSCH nodes.

A.1.3. Using BIER in a 6TiSCH Network

ROLL is actively working on Bit Index Explicit Replication (BIER) as a method to compress both the dataplane packets and the routing tables in storing mode [I-D.thubert-roll-bier].

BIER could also be used in the context of the DetNet service layer. BIER-TE-based OAM, Replication and Elimination [I-D.thubert-bier-replication-elimination] leverages BIER Traffic Engineering (TE) to control in the data plane the DetNet Replication and Elimination activities, and to provide traceability on links where replication and loss happen, in a manner that is abstract to the forwarding information.

a 6loRH for BitStrings [I-D.thubert-6lo-bier-dispatch] proposes a 6LoWPAN compression for the BIER Bitstring based on 6LoWPAN Routing Header [RFC8138].

A.2. External (non-IETF) work items

The current charter positions 6TiSCH on IEEE Std. 802.15.4 only. Though most of the design should be portable on other link types, 6TiSCH has a strong dependency on IEEE Std. 802.15.4 and its evolution. The impact of changes to TSCH on this Architecture should be minimal to non-existent, but deeper work such as 6top and security may be impacted. A 6TiSCH Interest Group at the IEEE maintains the synchronization and helps foster work at the IEEE should 6TiSCH demand it.

Work is being proposed at IEEE (802.15.12 PAR) for an LLC that would logically include the 6top sublayer. The interaction with the 6top sublayer and the Scheduling Functions described in this document are yet to be defined.

ISA100 [ISA100] Common Network Management (CNM) is another external work of interest for 6TiSCH. The group, referred to as ISA100.20, defines a Common Network Management framework that should enable the management of resources that are controlled by heterogeneous protocols such as ISA100.11a [ISA100.11a], WirelessHART [WirelessHART], and 6TiSCH. Interestingly, the establishment of 6TiSCH Deterministic paths, called Tracks, are also in scope, and ISA100.20 is working on requirements for DetNet.

Author's Address

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
06254 Mougins - Sophia Antipolis
France

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

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

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

December 10, 2019

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on June 12, 2020.

Table of Contents

1. Introduction

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

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

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

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

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

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

2. Terminology

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

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

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

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

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

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

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

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

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

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

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

3. Provisioning Phase

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

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

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

- pledge identifier. The pledge identifier identifies the (6LBR) pledge. The pledge identifier MUST be unique in the set of all pledge identifiers managed by a JRC. The pledge identifier uniqueness is an important security requirement, as discussed in Section 9. The pledge identifier is typically the globally unique 64-bit Extended Unique Identifier (EUI-64) of the IEEE Std 802.15.4 device, in which case it is provisioned by the hardware manufacturer. The pledge identifier is used to generate the IPv6 addresses of the (6LBR) pledge and to identify it during the execution of the join protocol. Depending on the configuration, the pledge identifier may also be used after the join process to identify the joined node. For privacy reasons (see Section 10), it is possible to use a pledge identifier different from the EUI-64. For example, a pledge identifier may be a random byte string, but care needs to be taken that such a string meets the uniqueness requirement.

- Pre-Shared Key (PSK). A symmetric cryptographic key shared between the (6LBR) pledge and the JRC. To look up the PSK for a given pledge, the JRC additionally needs to store the corresponding pledge identifier. Each (6LBR) pledge MUST be provisioned with a unique PSK. The PSK MUST be a cryptographically strong key, with at least 128 bits of entropy, indistinguishable by feasible computation from a random uniform string of the same length. How the PSK is generated and/or provisioned is out of scope of this specification. This could be done during a provisioning step or companion documents can specify the use of a key agreement protocol. Common pitfalls when generating PSKs are discussed in Section 9. In case of device re-commissioning to a new owner, the PSK MUST be changed. Note that the PSK is different from the link-layer keys K1 and K2 specified in [RFC8180]. The PSK is a long-term secret used to protect the execution of the secure join protocol specified in this document whose one output are link-layer keys.

- Optionally, a network identifier. The network identifier identifies the 6TiSCH network. The network identifier MUST be carried within Enhanced Beacon (EB) frames. Typically, the 16-bit Personal Area Network Identifier (PAN ID) defined in [IEEE802.15.4] is used as the network identifier. However, PAN ID is not considered a stable network identifier as it may change during network lifetime if a collision with another network is detected. Companion documents can specify the use of a different network identifier for join purposes, but this is out of scope of this specification. Provisioning the network identifier to a pledge is RECOMMENDED. However, due to operational constraints, the network identifier may not be known at the time when the provisioning is done. In case this parameter is not provisioned

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

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

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

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

4. Join Process Overview

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

1. The pledge listens for an Enhanced Beacon (EB) frame [IEEE802.15.4]. This frame provides network synchronization information, and tells the device when it can send a frame to the node sending the beacons, which acts as a Join Proxy (JP) for the pledge, and when it can expect to receive a frame. The Enhanced Beacon provides the link-layer address of the JP and it may also provide its link-local IPv6 address.

2. The pledge configures its link-local IPv6 address and advertises it to the JP using Neighbor Discovery. The advertisement step may be omitted if the link-local address has been derived from a known unique interface identifier, such as an EUI-64 address.

3. The pledge sends a Join Request to the JP in order to securely identify itself to the network. The Join Request is forwarded to the JRC.

4. In case of successful processing of the request, the pledge receives a Join Response from the JRC (via the JP). The Join Response contains configuration parameters necessary for the pledge to join the network.

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

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

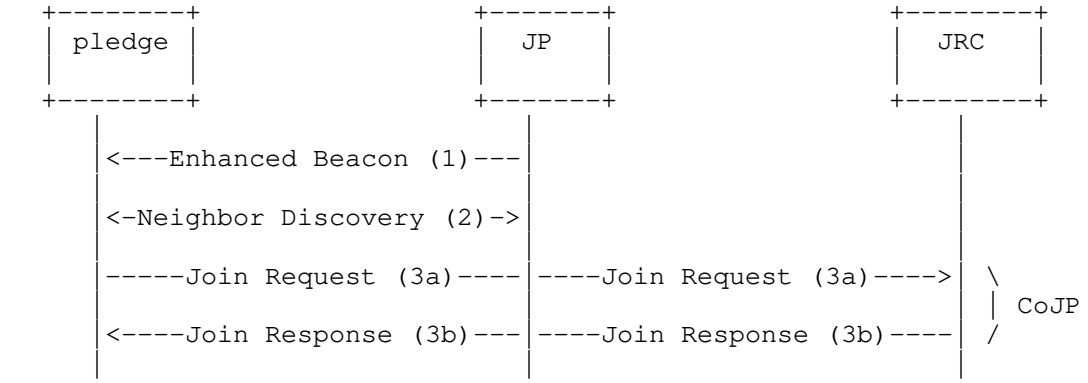

Figure 1: Overview of a successful join process.

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

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

4.1. Step 1 - Enhanced Beacon

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

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

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

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

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

4.2. Step 2 - Neighbor Discovery

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

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

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

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

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

4.3.1. Step 3a - Join Request

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

4.3.2. Step 3b - Join Response

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

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

4.4. The Special Case of the 6LBR Pledge Joining

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

5. Link-layer Configuration

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

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

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

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

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

5.1. Distribution of Time

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

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

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

6. Network-layer Configuration

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

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

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

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

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

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

6.1. Identification of Unauthenticated Traffic

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

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

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

6.1.1. Traffic from JP to JRC

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

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

6.1.2. Traffic from JRC to JP

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

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

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

7. Application-level Configuration

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

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

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

7.1. Statelessness of the JP

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

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

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

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

7.2. Recommended Settings

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

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

Recommended CoAP settings.

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

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

7.3. OSCORE

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

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

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

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

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

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

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

7.3.1. Replay Window and Persistency

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

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

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

7.3.2. OSCORE Error Handling

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

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

7.3.3. Mandatory to Implement Algorithms

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

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

8. Constrained Join Protocol (CoJP)

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

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

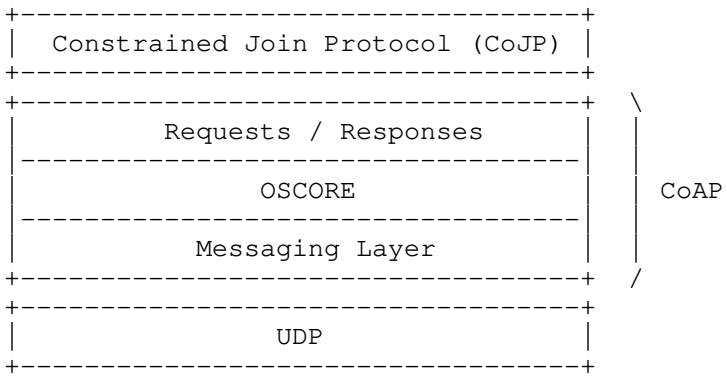

Figure 2: Abstract layering of CoJP.

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

- the Join Request message, sent by the (6LBR) pledge to the JRC, potentially proxied by the JP. The Join Request message and its mapping to CoAP is specified in Section 8.1.1.

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

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

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

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

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

8.1. Join Exchange

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

8.1.1. Join Request Message

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

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

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

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

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

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

8.1.2. Join Response Message

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

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

8.2. Parameter Update Exchange

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

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

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

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

8.2.1. Parameter Update Message

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

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

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

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

8.3. Error Handling

8.3.1. CoJP CBOR Object Processing

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

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

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

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

8.3.2. Diagnostic Response Message

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

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

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

8.3.3. Failure Handling

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

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

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

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

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

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

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

8.4. CoJP Objects

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

8.4.1. Join Request Object

The Join_Request structure is built on a CBOR map object.

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

- role: The identifier of the role that the pledge requests to play in the network once it joins, encoded as an unsigned integer. Possible values are specified in Table 2. This parameter MAY be included. In case the parameter is omitted, the default value of 0, i.e. the role "6TiSCH Node", MUST be assumed.

- network identifier: The identifier of the network, as discussed in Section 3, encoded as a CBOR byte string. When present in the Join_Request, it hints to the JRC the network that the pledge is requesting to join, enabling the JRC to manage multiple networks. The pledge obtains the value of the network identifier from the received EB frames. This parameter MUST be included in a Join_Request object regardless of the role parameter value.

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

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

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

Table 1: Summary of Join_Request parameters.

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

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

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

Table 2: Role values.

8.4.2. Configuration Object

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

- link-layer key set: An array encompassing a set of cryptographic keys and their identifiers that are currently in use in the network, or that are scheduled to be used in the future. The encoding of individual keys is described in Section 8.4.3. The link-layer key set parameter MAY be included in a Configuration object. When present, the link-layer key set parameter MUST contain at least one key. This parameter is also used to implement rekeying in the network. How the keys are installed and used differs for the 6LBR and other (regular) nodes, and this is explained in Section 8.4.3.1 and Section 8.4.3.2.

- short identifier: a compact identifier assigned to the pledge. The short identifier structure is described in Section 8.4.4. The short identifier parameter MAY be included in a Configuration object.

- JRC address: the IPv6 address of the JRC, encoded as a byte string, with the length of 16 bytes. If the length of the byte string is different from 16, the parameter MUST be discarded. If the JRC is not co-located with the 6LBR and has a different IPv6 address than the 6LBR, this parameter MUST be included. In the special case where the JRC is co-located with the 6LBR and has the same IPv6 address as the 6LBR, this parameter MAY be included. If the JRC address parameter is not present in the Configuration object, this indicates that the JRC has the same IPv6 address as the 6LBR. The joined node can then discover the IPv6 address of the JRC through network control traffic. See Section 6.

- blacklist: An array encompassing a list of pledge identifiers that are blacklisted by the JRC, with each pledge identifier encoded as a byte string. The blacklist parameter MAY be included in a Configuration object. When present, the array MUST contain zero or more byte strings encoding pledge identifiers. The joined node MUST silently drop any link-layer frames originating from the pledge identifiers enclosed in the blacklist parameter. When this parameter is received, its value MUST overwrite any previously set values. This parameter allows the JRC to configure the node acting as a JP to filter out traffic from misconfigured or malicious pledges before their traffic is forwarded into the network. If the JRC decides to remove a given pledge identifier from a blacklist, it omits the pledge identifier in the blacklist

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

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

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

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

Table 3: Summary of Configuration parameters.

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

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

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

Table 4: CoJP parameters map labels.

8.4.3. Link-Layer Key

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

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

For encoding compactness, the Link_Layer_Key object is not enclosed in a top-level CBOR object. Rather, it is transported as a sequence of CBOR elements [I-D.ietf-cbor-sequence], some being optional.

The set of parameters that can appear in a Link_Layer_Key object is summarized below, in order:

- key_id: The identifier of the key, encoded as a CBOR unsigned integer. This parameter MUST be included. If the decoded CBOR unsigned integer value is larger than the maximum link-layer key identifier, the key is considered invalid. In case the key is considered invalid, the key MUST be discarded and the implementation MUST signal the error as specified in Section 8.3.1.

- key_usage: The identifier of the link-layer algorithm, security level and link-layer frame types that can be used with the key, encoded as an integer. This parameter MAY be included. Possible values and the corresponding link-layer settings are specified in IANA "CoJP Key Usage" registry (Section 11.2). In case the parameter is omitted, the default value of 0 (6TiSCH-K1K2-ENC-MIC32) from Table 5 MUST be assumed. This default value has been chosen such that it results in byte savings in the most constrained settings but does not imply a recommendation for its general usage.

- key_value: The value of the cryptographic key, encoded as a byte string. This parameter MUST be included. If the length of the byte string is different than the corresponding key length for a given algorithm specified by the key_usage parameter, the key MUST be discarded and the implementation MUST signal the error as specified in Section 8.3.1.

- key_addinfo: Additional information needed to configure the link-layer key, encoded as a byte string. This parameter MAY be included. The processing of this parameter is dependent on the link-layer technology in use and a particular keying mode.

To be able to decode the keys that are present in the link-layer key set, and to identify individual parameters of a single Link_Layer_Key object, the CBOR decoder needs to differentiate between elements based on the CBOR type. For example, a uint that follows a byte string signals to the decoder that a new Link_Layer_Key object is being processed.

The CDDL fragment that represents the text above for the Link_Layer_Key follows.

```
Link_Layer_Key = (
      key_id             : uint,
    ? key_usage          : int,
      key_value          : bstr,
    ? key_addinfo        : bstr,
)
```

Name	Value	Algorithm	Description	Reference
6TiSCH-K1K2-ENC-MIC32	0	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, ENC-MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC64	1	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-ENC-MIC128	2	IEEE802154-AES-CCM-128	Use MIC-128 for EBs, ENC-MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC32	3	IEEE802154-AES-CCM-128	Use MIC-32 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K1K2-MIC64	4	IEEE802154-AES-CCM-128	Use MIC-64 for EBs, DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-	5	IEEE802154-AES-	Use MIC-128	[[this d

K1K2-MIC128		CCM-128	for EBs, DATA and ACKNOWLEDGMENT.	ocument]]
6TiSCH-K1-MIC32	6	IEEE802154-AES-CCM-128	Use MIC-32 for EBs.	[[this document]]
6TiSCH-K1-MIC64	7	IEEE802154-AES-CCM-128	Use MIC-64 for EBs.	[[this document]]
6TiSCH-K1-MIC128	8	IEEE802154-AES-CCM-128	Use MIC-128 for EBs.	[[this document]]
6TiSCH-K2-MIC32	9	IEEE802154-AES-CCM-128	Use MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-MIC64	10	IEEE802154-AES-CCM-128	Use MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-MIC128	11	IEEE802154-AES-CCM-128	Use MIC-128 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC32	12	IEEE802154-AES-CCM-128	Use ENC-MIC-32 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC64	13	IEEE802154-AES-CCM-128	Use ENC-MIC-64 for DATA and ACKNOWLEDGMENT.	[[this document]]
6TiSCH-K2-ENC-MIC128	14	IEEE802154-AES-CCM-128	Use ENC-MIC-128 for DATA and ACKNOWLEDGMEN	[[this document]]

| | | | | T. | | |
+-----------------+-----+------------------+-------------+----------+

Table 5: Key Usage values.

8.4.3.1. Rekeying of (6LoWPAN) Border Routers (6LBR)

When the 6LoWPAN Border Router (6LBR) receives the Configuration object containing a link-layer key set, it MUST immediately install and start using the new keys for all outgoing traffic, and remove any old keys it has installed from the previous key set after a delay of COJP_REKEYING_GUARD_TIME has passed. This mechanism is used by the JRC to force the 6LBR to start sending traffic with the new key. The decision is taken by the JRC when it has determined that the new key has been made available to all (or some overwhelming majority) of nodes. Any node that the JRC has not yet reached at that point is either non-functional or in extended sleep such that it will not be reached. To get the key update, such node needs to go through the join process anew.

8.4.3.2. Rekeying of regular (6LoWPAN) Nodes (6LN)

When a regular 6LN node receives the Configuration object with a link-layer key set, it MUST install the new keys. The 6LN will use both the old and the new keys to decrypt and authenticate any incoming traffic that arrives based upon the key identifier in the packet. It MUST continue to use the old keys for all outgoing traffic until it has detected that the network has switched to the new key set.

The detection of network switch is based upon the receipt of traffic secured with the new keys. Upon reception and successful security processing of a link-layer frame secured with a key from the new key set, a 6LN node MUST then switch to sending outgoing traffic using the keys from the new set for all outgoing traffic. The 6LN node MUST remove any old keys it has installed from the previous key set after a delay of COJP_REKEYING_GUARD_TIME has passed after it starts using the new key set.

Sending of traffic with the new keys signals to other downstream nodes to switch to their new key, and the effect is that there is a ripple of key updates around each 6LBR.

8.4.3.3. Use in IEEE Std 802.15.4

When Link_Layer_Key is used in the context of [IEEE802.15.4], the following considerations apply.

Signaling of different keying modes of [IEEE802.15.4] is done based on the parameter values present in a Link_Layer_Key object. For instance, the value of the key_id parameter in combination with key_addinfo denotes which of the four Key ID modes of [IEEE802.15.4] is used and how.

- Key ID Mode 0x00 (Implicit, pairwise): key_id parameter MUST be set to 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to the link-layer address(es) of a single peer with whom the key should be used. Depending on the configuration of the network, key_addinfo may carry the peer's long link-layer address (i.e. pledge identifier), short link-layer address, or their concatenation with the long address being encoded first. Which address type(s) is carried is determined from the length of the byte string.

- Key ID Mode 0x01 (Key Index): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST NOT be present.

- Key ID Mode 0x02 (4-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 4 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].

- Key ID Mode 0x03 (8-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 8 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].

In all cases, key_usage parameter determines how a particular key should be used in respect to incoming and outgoing security policies.

For Key ID Modes 0x01 - 0x03, parameter key_id sets the "secKeyIndex" parameter of [IEEE802.15.4] that is signaled in all outgoing frames secured with a given key. The maximum value key_id can have is 254. The value of 255 is reserved in [IEEE802.15.4] and is therefore considered invalid.

Key ID Mode 0x00 (Implicit, pairwise) enables the JRC to act as a trusted third party and assign pairwise keys between nodes in the network. How JRC learns about the network topology is out of scope of this specification, but could be done through 6LBR - JRC signaling for example. Pairwise keys could also be derived through a key agreement protocol executed between the peers directly, where the authentication is based on the symmetric cryptographic material

provided to both peers by the JRC. Such a protocol is out of scope of this specification.

Implementations MUST use different link-layer keys when using different authentication tag (MIC) lengths, as using the same key with different authentication tag lengths might be unsafe. For example, this prohibits the usage of the same key for both MIC-32 and MIC-64 levels. See Annex B.4.3 of [IEEE802.15.4] for more information.

8.4.4. Short Identifier

The Short_Identifier object represents an identifier assigned to the pledge. It is encoded as a CBOR array object, containing, in order:

- identifier: The short identifier assigned to the pledge, encoded as a byte string. This parameter MUST be included. The identifier MUST be unique in the set of all identifiers assigned in a network that is managed by a JRC. In case the identifier is invalid, the decoder MUST silently ignore the Short_Identifier object.

- lease_time: The validity of the identifier in hours after the reception of the CBOR object, encoded as a CBOR unsigned integer. This parameter MAY be included. The node MUST stop using the assigned short identifier after the expiry of the lease_time interval. It is up to the JRC to renew the lease before the expiry of the previous interval. The JRC updates the lease by executing the Parameter Update exchange with the node and including the Short_Identifier in the Configuration object, as described in Section 8.2. In case the lease expires, the node SHOULD initiate a new join exchange, as described in Section 8.1. In case this parameter is omitted, the value of positive infinity MUST be assumed, meaning that the identifier is valid for as long as the node participates in the network.

The CDDL fragment that represents the text above for the Short_Identifier follows.

```
Short_Identifier = [
      identifier        : bstr,
    ? lease_time        : uint
]
```

8.4.4.1. Use in IEEE Std 802.15.4

When Short_Identifier is used in the context of [IEEE802.15.4], the following considerations apply.

The identifier MUST be used to set the short address of IEEE Std 802.15.4 module. When operating in TSCH mode, the identifier MUST be unique in the set of all identifiers assigned in multiple networks that share link-layer key(s). If the length of the byte string corresponding to the identifier parameter is different than 2, the identifier is considered invalid. The values 0xfffe and 0xffff are reserved by [IEEE802.15.4] and their use is considered invalid.

The security properties offered by the [IEEE802.15.4] link-layer in TSCH mode are conditioned on the uniqueness requirement of the short identifier (i.e. short address). The short address is one of the inputs in the construction of the nonce, which is used to protect link-layer frames. If a misconfiguration occurs, and the same short address is assigned twice under the same link-layer key, the loss of security properties is imminent. For this reason, practices where the pledge generates the short identifier locally are not safe and are likely to result in the loss of link-layer security properties.

The JRC MUST ensure that at any given time there are never two same short identifiers being used under the same link-layer key. If the lease_time parameter of a given Short_Identifier object is set to positive infinity, care needs to be taken that the corresponding identifier is not assigned to another node until the JRC is certain that it is no longer in use, potentially through out-of-band signaling. If the lease_time parameter expires for any reason, the JRC should take into consideration potential ongoing transmissions by the joined node, which may be hanging in the queues, before assigning the same identifier to another node.

Care needs to be taken on how the pledge (joined node) configures the expiration of the lease. Since units of the lease_time parameter are in hours after the reception of the CBOR object, the pledge needs to convert the received time to the corresponding absolute slot number in the network. The joined node (pledge) MUST only use the absolute slot number as the appropriate reference of time to determine whether the assigned short identifier is still valid.

8.4.5. Unsupported Configuration Object

The Unsupported_Configuration object is encoded as a CBOR array, containing at least one Unsupported_Parameter object. Each Unsupported_Parameter object is a sequence of CBOR elements without an enclosing top-level CBOR object for compactness. The set of

parameters that appear in an Unsupported_Parameter object is summarized below, in order:

- code: Indicates the capability of acting on the parameter signaled by parameter_label, encoded as an integer. This parameter MUST be included. Possible values of this parameter are specified in the IANA "CoJP Unsupported Configuration Code Registry" (Section 11.3).

- parameter_label: Indicates the parameter. This parameter MUST be included. Possible values of this parameter are specified in the label column of the IANA "CoJP Parameters" registry (Section 11.1).

- parameter_addinfo: Additional information about the parameter that cannot be acted upon. This parameter MUST be included. In case the code is set to "Unsupported", parameter_addinfo gives additional information to the JRC. If the parameter indicated by parameter_label cannot be acted upon regardless of its value, parameter_addinfo MUST be set to null, signaling to the JRC that it SHOULD NOT attempt to configure the parameter again. If the pledge can act on the parameter, but cannot configure the setting indicated by the parameter value, the pledge can hint this to the JRC. In this case, parameter_addinfo MUST be set to the value of the parameter that cannot be acted upon following the normative parameter structure specified in this document. For example, it is possible to include the link-layer key set object, signaling a subset of keys that cannot be acted upon, or the entire key set that was received. In that case, the value of the parameter_addinfo follows the link-layer key set structure defined in Section 8.4.2. In case the code is set to "Malformed", parameter_addinfo MUST be set to null, signaling to the JRC that it SHOULD NOT attempt to configure the parameter again.

The CDDL fragment that represents the text above for Unsupported_Configuration and Unsupported_Parameter objects follows.

```
Unsupported_Configuration = [
       + parameter           : Unsupported_Parameter
]

Unsupported_Parameter = (
         code                : int,
         parameter_label     : int,
         parameter_addinfo   : nil / any
)
```

Name	Value	Description	Reference
Unsupported	0	The indicated setting is not supported by the networking stack implementation.	[[this document]]
Malformed	1	The indicated parameter value is malformed.	[[this document]]

Table 6: Unsupported Configuration code values.

8.5. Recommended Settings

This section gives RECOMMENDED values of CoJP settings.

Name	Default Value
COJP_MAX_JOIN_ATTEMPTS	4
COJP_REKEYING_GUARD_TIME	12 seconds

Recommended CoJP settings.

The COJP_REKEYING_GUARD_TIME value SHOULD take into account possible retransmissions at the link layer due to imperfect wireless links.

9. Security Considerations

Since this document uses the pledge identifier to set the ID Context parameter of OSCORE, an important security requirement is that the pledge identifier is unique in the set of all pledge identifiers managed by a JRC. The uniqueness of the pledge identifier ensures unique (key, nonce) pairs for AEAD algorithm used by OSCORE. It also allows the JRC to retrieve the correct security context, upon the reception of a Join Request message. The management of pledge identifiers is simplified if the globally unique EUI-64 is used, but this comes with privacy risks, as discussed in Section 10.

This document further mandates that the (6LBR) pledge and the JRC are provisioned with unique PSKs. While the process of provisioning PSKs to all pledges can result in a substantial operational overhead, it is vital to do so for the security properties of the network. The PSK is used to set the OSCORE Master Secret during security context derivation. This derivation process results in OSCORE keys that are

important for mutual authentication of the (6LBR) pledge and the JRC. The resulting security context shared between the pledge (joined node) and the JRC is used for the purpose of joining and is long-lived in that it can be used throughout the lifetime of a joined node for parameter update exchanges. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible.

Note that while OSCORE provides replay protection, it does not provide an indication of freshness in the presence of an attacker that can drop/reorder traffic. Since the join request contains no randomness, and the sequence number is predictable, the JRC could in principle anticipate a join request from a particular pledge and pre-calculate the response. In such a scenario, the JRC does not have to be alive at the time when the request is received. This could be relevant in case the JRC was temporarily compromised and control subsequently regained by the legitimate owner.

It is of utmost importance to avoid unsafe practices when generating and provisioning PSKs. The use of a single PSK shared among a group of devices is a common pitfall that results in poor security. In this case, the compromise of a single device is likely to lead to a compromise of the entire batch, with the attacker having the ability to impersonate a legitimate device and join the network, generate bogus data and disturb the network operation. Additionally, some vendors use methods such as scrambling or hashing of device serial numbers or their EUI-64 to generate "unique" PSKs. Without any secret information involved, the effort that the attacker needs to invest into breaking these unsafe derivation methods is quite low, resulting in the possible impersonation of any device from the batch, without even needing to compromise a single device. The use of cryptographically secure random number generators to generate the PSK is RECOMMENDED, see [NIST800-90A] for different mechanisms using deterministic methods.

The JP forwards the unauthenticated join traffic into the network. A data cap on the JP prevents it from forwarding more traffic than the network can handle and enables throttling in case of an attack. Note that this traffic can only be directed at the JRC so that the JRC needs to be prepared to handle such unsanitized inputs. The data cap can be configured by the JRC by including a join rate parameter in the Join Response and it is implemented through the CoAP's PROBING_RATE setting. The use of a data cap at a JP forces attackers to use more than one JP if they wish to overwhelm the network. Marking the join traffic packets with a non-zero DSCP allows the network to carry the traffic if it has capacity, but encourages the network to drop the extra traffic rather than add bandwidth due to that traffic.

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

At the beginning of the join process, the pledge has no means of verifying the content in the EB, and has to accept it at "face value". In case the pledge tries to join an attacker's network, the Join Response message will either fail the security check or time out. The pledge may implement a temporary blacklist in order to filter out undesired EBs and try to join using the next seemingly valid EB. This blacklist alleviates the issue, but is effectively limited by the node's available memory. Note that this temporary blacklist is different from the one communicated as part of the CoJP Configuration object as it helps pledge fight a DoS attack. The bogus beacons prolong the join time of the pledge, and so the time spent in "minimal" [RFC8180] duty cycle mode. The blacklist communicated as part of the CoJP Configuration object helps JP fight a DoS attack by a malicious pledge.

During the network lifetime, the JRC may at any time initiate a Parameter Update exchange with a joined node. The Parameter Update message uses the same OSCORE security context as is used for the join exchange, except that the server/client roles are interchanged. As a consequence, each Parameter Update message carries the well-known OSCORE Sender ID of the JRC. A passive attacker may use the OSCORE Sender ID to identify the Parameter Update traffic in case the link-layer protection does not provide confidentiality. A countermeasure against such traffic analysis attack is to use encryption at the link-layer. Note that the join traffic does not undergo link-layer protection at the first hop, as the pledge is not yet in possession of cryptographic keys. Similarly, enhanced beacon traffic in the network is not encrypted. This makes it easy for a passive attacker to identify these types of traffic.

10. Privacy Considerations

The join solution specified in this document relies on the uniqueness of the pledge identifier in the set of all pledge identifiers managed by a JRC. This identifier is transferred in clear as an OSCORE kid context. The use of the globally unique EUI-64 as pledge identifier simplifies the management but comes with certain privacy risks. The implications are thoroughly discussed in [RFC7721] and comprise correlation of activities over time, location tracking, address scanning and device-specific vulnerability exploitation. Since the

join process occurs rarely compared to the network lifetime, long-term threats that arise from using EUI-64 as the pledge identifier are minimal. However, the use of EUI-64 after the join process completes, in the form of a layer-2 or layer-3 address, extends the aforementioned privacy threats to long term.

As an optional mitigation technique, the Join Response message may contain a short address which is assigned by the JRC to the (6LBR) pledge. The assigned short address SHOULD be uncorrelated with the long-term pledge identifier. The short address is encrypted in the response. Once the join process completes, the new node may use the short addresses for all further layer-2 (and layer-3) operations. This reduces the privacy threats as the short layer-2 address (visible even when the network is encrypted) does not disclose the manufacturer, as is the case of EUI-64. However, an eavesdropper with access to the radio medium during the join process may be able to correlate the assigned short address with the extended address based on timing information with a non-negligible probability. This probability decreases with an increasing number of pledges joining concurrently.

11. IANA Considerations

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

This document allocates a well-known name under the .arpa name space according to the rules given in [RFC3172]. The name "6tisch.arpa" is requested. No subdomains are expected, and addition of any such subdomains requires the publication of an IETF standards-track RFC. No A, AAAA or PTR record is requested.

11.1. CoJP Parameters Registry

This section defines a sub-registry within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Parameters Registry".

The columns of the registry are:

Name: This is a descriptive name that enables an easier reference to the item. It is not used in the encoding.

Label: The value to be used to identify this parameter. The label is an integer.

CBOR type: This field contains the CBOR type for the field.

Description: This field contains a brief description for the field.

Reference: This field contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 4.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

11.2. CoJP Key Usage Registry

This section defines a sub-registry within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Key Usage Registry".

The columns of this registry are:

Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

Value: This is the value used to identify the key usage setting. These values MUST be unique. The value is an integer.

Algorithm: This is a descriptive name of the link-layer algorithm in use and uniquely determines the key length. The name is not used in the encoding.

Description: This field contains a description of the key usage setting. The field should describe in enough detail how the key is to be used with different frame types, specific for the link-layer technology in question.

Reference: This contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 5.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are

designated as Expert Review. Integer values less than -65536 are marked as Private Use.

11.3. CoJP Unsupported Configuration Code Registry

This section defines a sub-registry within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry with the name "Constrained Join Protocol Unsupported Configuration Code Registry".

The columns of this registry are:

Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

Value: This is the value used to identify the diagnostic code. These values MUST be unique. The value is an integer.

Description: This is a descriptive human-readable name. The description MUST be unique. It is not used in the encoding.

Reference: This contains a pointer to the public specification for the field, if one exists.

This registry is to be populated with the values in Table 6.

The amending formula for this sub-registry is: Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

12. Acknowledgments

The work on this document has been partially supported by the European Union's H2020 Programme for research, technological development and demonstration under grant agreements: No 644852, project ARMOUR; No 687884, project F-Interop and open-call project SPOTS; No 732638, project Fed4FIRE+ and open-call project SODA.

The following individuals provided input to this document (in alphabetic order): Christian Amsuss, Tengfei Chang, Klaus Hartke, Tero Kivinen, Jim Schaad, Goeran Selander, Yasuyuki Tanaka, Pascal Thubert, William Vignat, Xavier Vilajosana, Thomas Watteyne.

13. References

13.1. Normative References

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-28 (work in progress), October 2019.

[I-D.ietf-core-stateless]
Hartke, K., "Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)", draft-ietf-core-stateless-03 (work in progress), October 2019.

[IEEE802.15.4]
IEEE standard for Information Technology, ., "IEEE Std 802.15.4 Standard for Low-Rate Wireless Networks", n.d..

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

[RFC2597] Heinanen, J., Baker, F., Weiss, W., and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, DOI 10.17487/RFC2597, June 1999, <https://www.rfc-editor.org/info/rfc2597>.

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/info/rfc3172>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

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

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <https://www.rfc-editor.org/info/rfc7320>.

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

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

13.2. Informative References

[I-D.ietf-6tisch-msf]
Chang, T., Vucinic, M., Vilajosana, X., Duquennoy, S., and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)", draft-ietf-6tisch-msf-08 (work in progress), November 2019.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-15 (work in progress), July 2017.

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", draft-ietf-cbor-cddl-08 (work in progress), March 2019.

[I-D.ietf-cbor-sequence]
Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", draft-ietf-cbor-sequence-02 (work in progress), September 2019.

[NIST800-90A]
NIST Special Publication 800-90A, Revision 1, ., Barker, E., and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", 2015.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <https://www.rfc-editor.org/info/rfc4231>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <https://www.rfc-editor.org/info/rfc7721>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[RFC8480] Wang, Q., Ed., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)", RFC 8480, DOI 10.17487/RFC8480, November 2018, <https://www.rfc-editor.org/info/rfc8480>.

Appendix A. Example

Figure 3 illustrates a successful join protocol exchange. The pledge instantiates the OSCORE context and derives the OSCORE keys and nonces from the PSK. It uses the instantiated context to protect the Join Request addressed with a Proxy-Scheme option, the well-known host name of the JRC in the Uri-Host option, and its EUI-64 as pledge identifier and OSCORE kid context. Triggered by the presence of a Proxy-Scheme option, the JP forwards the request to the JRC and sets the CoAP token to the internally needed state. The JP has learned the IPv6 address of the JRC when it acted as a pledge and joined the network. Once the JRC receives the request, it looks up the correct context based on the kid context parameter. The OSCORE data authenticity verification ensures that the request has not been modified in transit. In addition, replay protection is ensured through persistent handling of mutable context parameters.

Once the JP receives the Join Response, it authenticates the state within the CoAP token before deciding where to forward. The JP sets its internal state to that found in the token, and forwards the Join Response to the correct pledge. Note that the JP does not possess

the key to decrypt the CoJP object (configuration) present in the payload. The Join Response is matched to the Join Request and verified for replay protection at the pledge using OSCORE processing rules. In this example, the Join Response does not contain the IPv6 address of the JRC, the pledge hence understands the JRC is co-located with the 6LBR.

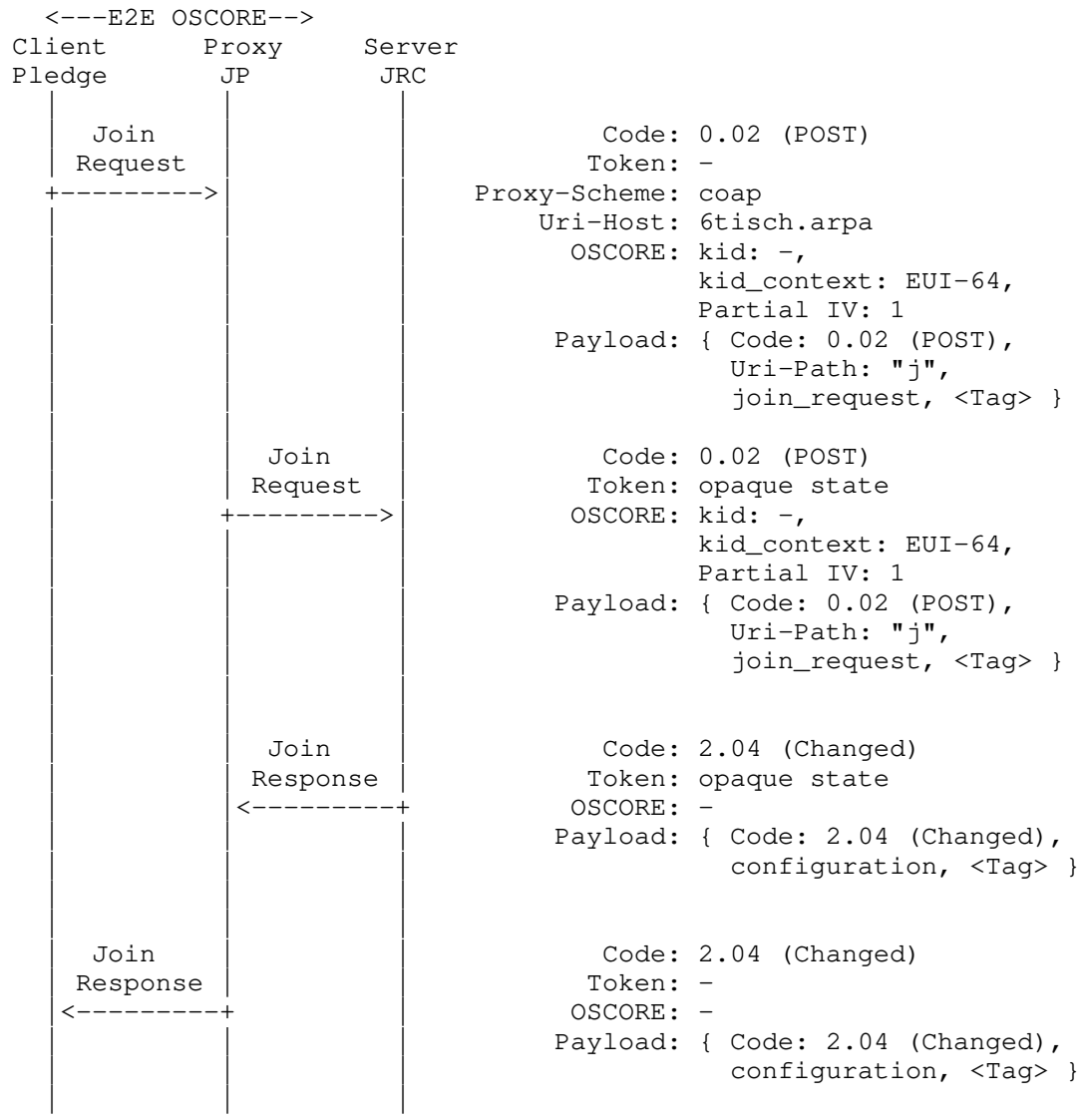

Figure 3: Example of a successful join protocol exchange. { ... } denotes authenticated encryption, <Tag> denotes the authentication tag.

Where the join_request object is:

```
join_request:
{
    5 : h'cafe' / PAN ID of the network pledge is attempting to join /
}
```

Since the role parameter is not present, the default role of "6TiSCH Node" is implied.

The join_request object encodes to h'a10542cafe' with a size of 5 bytes.

And the configuration object is:

```
configuration:
{
    2 : [           / link-layer key set /
          1,        / key_id /
          h'e6bf4287c2d7618d6a9687445ffd33e6' / key_value /
        ],
    3 : [           / short identifier /
          h'af93'   / assigned short address /
        ]
}
```

Since the key_usage parameter is not present in the link-layer key set object, the default value of "6TiSCH-K1K2-ENC-MIC32" is implied. Since key_addinfo parameter is not present and key_id is different than 0, Key ID Mode 0x01 (Key Index) is implied. Similarly, since the lease_time parameter is not present in the short identifier object, the default value of positive infinity is implied.

The configuration object encodes to

h'a202820150e6bf4287c2d7618d6a9687445ffd33e6038142af93' with a size of 26 bytes.

Appendix B. Lightweight Implementation Option

In environments where optimizing the implementation footprint is important, it is possible to implement this specification without having the implementations of HKDF [RFC5869] and SHA [RFC4231] on constrained devices. HKDF and SHA are used during the OSCORE security context derivation phase. This derivation can also be done by the JRC or a provisioning device, on behalf of the (6LBR) pledge during the provisioning phase. In that case, the derived OSCORE security context parameters are written directly into the (6LBR) pledge, without requiring the PSK be provisioned to the (6LBR) pledge.

The use of HKDF to derive OSCORE security context parameters ensures that the resulting OSCORE keys have good security properties, and are unique as long as the input for different pledges varies. This specification ensures the uniqueness by mandating unique pledge identifiers and a unique PSK for each (6LBR) pledge. From the AEAD nonce reuse viewpoint, having a unique pledge identifier is a sufficient condition. However, as discussed in Section 9, the use of a single PSK shared among many devices is a common security pitfall. The compromise of this shared PSK on a single device would lead to the compromise of the entire batch. When using the implementation/ deployment scheme outlined above, the PSK does not need to be written to individual pledges. As a consequence, even if a shared PSK is used, the scheme offers a comparable level of security as in the scenario where each pledge is provisioned with a unique PSK. In this case, there is still a latent risk of the shared PSK being compromised from the provisioning device, which would compromise all devices in the batch.

Authors' Addresses

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

Email: malisa.vucinic@inria.fr

Jonathan Simon
Analog Devices
32990 Alvarado-Niles Road, Suite 910
Union City, CA 94587
USA

Email: jonathan.simon@analog.com

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

Email: pister@eecs.berkeley.edu

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

Email: mcr+ietf@sandelman.ca

6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2021

T. Chang, Ed.
M. Vucinic
Inria
X. Vilajosana
Universitat Oberta de Catalunya
S. Duquennoy
RISE SICS
D. Dujovne
Universidad Diego Portales

September 12, 2020

6TiSCH Minimal Scheduling Function (MSF)
draft-ietf-6tisch-msf-18

Abstract

This specification defines the 6TiSCH Minimal Scheduling Function (MSF). This Scheduling Function describes both the behavior of a node when joining the network, and how the communication schedule is managed in a distributed fashion. MSF is built upon the 6TiSCH Operation Sublayer Protocol (6P) and the Minimal Security Framework for 6TiSCH.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 16, 2021.

Table of Contents

1. Introduction

The 6TiSCH Minimal Scheduling Function (MSF), defined in this specification, is a 6TiSCH Scheduling Function (SF). The role of an SF is entirely defined in [RFC8480]. This specification complements [RFC8480] by providing the rules of when to add/delete cells in the communication schedule. This specification satisfies all the requirements for an SF listed in Section 4.2 of [RFC8480].

MSF builds on top of the following specifications: the Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration [RFC8180], the 6TiSCH Operation Sublayer Protocol (6P) [RFC8480], and the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security].

MSF defines both the behavior of a node when joining the network, and how the communication schedule is managed in a distributed fashion. When a node running MSF boots up, it joins the network by following the 6 steps described in Section 4. The end state of the join process is that the node is synchronized to the network, has mutually authenticated with the network, has identified a routing parent, and has scheduled one negotiated Tx cell (defined in Section 5.1) to/from its routing parent. After the join process, the node can continuously add/delete/relocate cells, as described in Section 5. It does so for 3 reasons: to match the link-layer resources to the traffic, to handle changing parent and to handle a schedule collision.

MSF works closely with the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), specifically the routing parent defined in [RFC6550]. This specification only describes how MSF works with the routing parent; this parent is referred to as the "selected parent". The activity of MSF towards the single routing parent is called a "MSF session". Though the performance of MSF is evaluated only when the "selected parent" represents the node's preferred parent, there should be no restrictions to use multiple MSF sessions, one per parent. The distribution of traffic over multiple parents is a routing decision that is out of scope for MSF.

MSF is designed to operate in a wide range of application domains. It is optimized for applications with regular upstream traffic, from the nodes to the Destination-Oriented Directed Acyclic Graph (DODAG [RFC6550]) root.

This specification follows the recommended structure of an SF specification, given in Appendix A of [RFC8480], with the following adaptations:

* We have reordered some sections, in particular to have the section on the node behavior at boot (Section 4) appear early in this specification.
* We added sections on the interface to the minimal 6TiSCH configuration (Section 2), the use of the SIGNAL command (Section 6), the MSF constants (Section 14) and the MSF statistics (Section 15).

2. Interface to the Minimal 6TiSCH Configuration

In a TSCH network, time is sliced up into time slots. The time slots are grouped as one or multiple slotframes which repeat over time. The TSCH schedule instructs a node what to do at each time slot, such as transmit, receive or sleep [RFC7554]. In case of a slot to transmit or receive, a channel is assigned to the time slot. The tuple (slot, channel) is indicated as a cell of TSCH schedule. MSF is one of the policies defining how to manage the TSCH schedule.

A node implementing MSF SHOULD implement the Minimal 6TiSCH Configuration [RFC8180], which defines the "minimal cell", a single shared cell providing minimal connectivity between the nodes in the network. The MSF implementation provided in this specification is based on the implementation of the Minimal 6TiSCH Configuration. However, an implementor MAY implement MSF based on other specifications as long as the specification defines a way to advertise the EB/DIO among the network.

MSF uses the minimal cell for broadcast frames such as Enhanced Beacons (EBs) [IEEE802154] and broadcast DODAG Information Objects (DIOs) [RFC6550]. Cells scheduled by MSF are meant to be used only for unicast frames.

To ensure there is enough bandwidth available on the minimal cell, a node implementing MSF SHOULD enforce some rules for limiting the traffic of broadcast frames. For example, the overall broadcast traffic among the node and its neighbors SHOULD NOT exceed 1/3 of the bandwidth of minimal cell. One of the algorithms that fulfills this requirement is the Trickle timer defined in [RFC6206] which is applied on DIO messages [RFC6550]. However, any such algorithm of limiting the broadcast traffic to meet those rules is implementation-specific and is out of the scope of MSF.

3 slotframes are used in MSF. MSF schedules autonomous cells at Slotframe 1 (Section 3) and 6P negotiated cells at Slotframe 2 (Section 5) ,while Slotframe 0 is used for the bootstrap traffic as defined in the Minimal 6TiSCH Configuration. The same slotframe length for Slotframe 0, 1 and 2 is RECOMMENDED. Thus it is possible to avoid the scheduling collision between the autonomous cells and 6P negotiated cells (Section 3). The default slotframe length (SLOTFRAME_LENGTH) is RECOMMENDED for Slotframe 0, 1 and 2, although any value can be advertised in the EBs.

3. Autonomous Cells

MSF nodes initialize Slotframe 1 with a set of default cells for unicast communication with their neighbors. These cells are called 'autonomous cells', because they are maintained autonomously by each node without negotiation through 6P. Cells scheduled by 6P transaction are called 'negotiated cells' which are reserved on Slotframe 2. How to schedule negotiated cells is detailed in Section 5. There are two types of autonomous cells:

* Autonomous Rx Cell (AutoRxCell), one cell at a [slotOffset,channelOffset] computed as a hash of the EUI64 of the node itself (detailed next). Its cell options bits are assigned as TX=0, RX=1, SHARED=0.
* Autonomous Tx Cell (AutoTxCell), one cell at a [slotOffset,channelOffset] computed as a hash of the layer 2 EUI64 destination address in the unicast frame to be transmitted (detailed in Section 4.4). Its cell options bits are assigned as TX=1, RX=0, SHARED=1.

To compute a [slotOffset,channelOffset] from an EUI64 address, nodes MUST use the hash function SAX as defined in Section 2 of [SAX-DASFAA] with consistent input parameters, for example, those defined in Appendix A. The coordinates are computed to distribute the cells across all channel offsets, and all but the first slot offset of Slotframe 1. The first time offset is skipped to avoid colliding with the minimal cell in Slotframe 0. The slot coordinates derived from a given EUI64 address are computed as follows:

* slotOffset(MAC) = 1 + hash(EUI64, length(Slotframe_1) - 1)
* channelOffset(MAC) = hash(EUI64, NUM_CH_OFFSET)

The second input parameter defines the maximum return value of the hash function. Other optional parameters defined in SAX determine the performance of SAX hash function. Those parameters could be broadcasted in EB frame or pre-configured. For interoperability purposes, the values of those parameters can be referred from Appendix A.

AutoTxCell is not permanently installed in the schedule but added/deleted on demand when there is a frame to be sent. Throughout the network lifetime, nodes maintain the autonomous cells as follows:

* Add an AutoTxCell to the layer 2 destination address which is indicated in a frame when there is no 6P negotiated Tx cell in schedule for that frame to transmit.
* Remove an AutoTxCell when:
  - there is no frame to transmit on that cell, or
  - there is at least one 6P negotiated Tx cell in the schedule for the frames to transmit.

The AutoRxCell MUST always remain scheduled after synchronization. 6P CLEAR MUST NOT erase any autonomous cells.

Because of hash collisions, there will be cases that the AutoTxCell and AutoRxCell are scheduled at the same slot offset and/or channel offset. In such cases, AutoTxCell always take precedence over AutoRxCell. Notice AutoTxCell is a shared type cell which applies backs-off mechanism. When the AutoTxCell and AutoRxCell collide, AutoTxCell takes precedence if there is a packet to transmit. When in a back-off period, AutoRxCell is used. In case of conflicting with a negotiated cell, autonomous cells take precedence over negotiated cells, which is stated in [IEEE802154]. However, when the Slotframe 0, 1 and 2 use the same length value, it is possible for a negotiated cell to avoid the collision with AutoRxCell. Hence, the same slotframe length for Slotframe 0, 1 and 2 is RECOMMENDED.

4. Node Behavior at Boot

This section details the behavior the node SHOULD follow from the moment it is switched on, until it has successfully joined the network. Alternative behaviors may be involved, for example, when alternative security solutions are used for the network. Section 4.1 details the start state; Section 4.8 details the end state. The other sections detail the 6 steps of the joining process. We use the term "pledge" and "joined node", as defined in [I-D.ietf-6tisch-minimal-security].

4.1. Start State

A node implementing MSF SHOULD implement the Constrained Join Protocol (CoJP) for 6TiSCH [I-D.ietf-6tisch-minimal-security]. As a corollary, this means that a pledge, before being switched on, may be pre-configured with the Pre-Shared Key (PSK) for joining, as well as any other configuration detailed in ([I-D.ietf-6tisch-minimal-security]). This is not necessary if the

node implements a security solution not based on PSKs, such as ([I-D.ietf-6tisch-dtsecurity-zerotouch-join]).

4.2. Step 1 - Choosing Frequency

When switched on, the pledge randomly chooses a frequency from the channels that the network cycles amongst, and starts listening for EBs on that frequency.

4.3. Step 2 - Receiving EBs

Upon receiving the first EB, the pledge continues listening for additional EBs to learn:

1. the number of neighbors N in its vicinity
2. which neighbor to choose as a Join Proxy (JP) for the joining process

After having received the first EB, a node MAY keep listening for at most MAX_EB_DELAY seconds or until it has received EBs from NUM_NEIGHBOURS_TO_WAIT distinct neighbors. This behavior is defined in [RFC8180].

During this step, the pledge only gets synchronized when it received enough EB from the network it wishes to join. How to decide whether an EB originates from a node from the network it wishes to join is implementation-specific, but MAY involve filtering EBs by the PAN ID field it contains, the presence and contents of the IE defined in [I-D.ietf-6tisch-enrollment-enhanced-beacon], or the key used to authenticate it.

The decision of which neighbor to use as a JP is implementation-specific, and discussed in [I-D.ietf-6tisch-minimal-security].

4.4. Step 3 - Setting up Autonomous Cells for the Join Process

After having selected a JP, a node generates a Join Request and installs an AutoTxCell to the JP. The Join Request is then sent by the pledge to its selected JP over the AutoTxCell. The AutoTxCell is removed by the pledge when the Join Request is sent out. The JP receives the Join Request through its AutoRxCell. Then it forwards the Join Request to the join registrar/coordinator (JRC), possibly over multiple hops, over the 6P negotiated Tx cells. Similarly, the JRC sends the Join Response to the JP, possibly over multiple hops, over AutoTxCells or the 6P negotiated Tx cells. When the JP received the Join Response from the JRC, it installs an AutoTxCell to the pledge and sends that Join Response to the pledge over AutoTxCell. The AutoTxCell is removed by the JP when the Join Response is sent

out. The pledge receives the Join Response from its AutoRxCell, thereby learns the keying material used in the network, as well as other configuration settings, and becomes a "joined node".

When 6LoWPAN Neighbor Discovery ([RFC8505]) (ND) is implemented, the unicast packets used by ND are sent on the AutoTxCell. The specific process how the ND works during the Join process is detailed in [I-D.ietf-6tisch-architecture].

4.5. Step 4 - Acquiring a RPL Rank

Per [RFC6550], the joined node receives DIOs, computes its own Rank, and selects a routing parent.

4.6. Step 5 - Setting up first Tx negotiated Cells

Once it has selected a routing parent, the joined node MUST generate a 6P ADD Request and install an AutoTxCell to that parent. The 6P ADD Request is sent out through the AutoTxCell, containing the following fields:

* CellOptions: set to TX=1,RX=0,SHARED=0
* NumCells: set to 1
* CellList: at least 5 cells, chosen according to Section 8

The joined node removes the AutoTxCell to the selected parent when the 6P Request is sent out. That parent receives the 6P ADD Request from its AutoRxCell. Then it generates a 6P ADD Response and installs an AutoTxCell to the joined node. When the parent sends out the 6P ADD Response, it MUST remove that AutoTxCell. The joined node receives the 6P ADD Response from its AutoRxCell and completes the 6P transaction. In case the 6P ADD transaction failed, the node MUST issue another 6P ADD Request and repeat until the Tx cell is installed to the parent.

4.7. Step 6 - Send EBs and DIOs

The node starts sending EBs and DIOs on the minimal cell, while following the transmit rules for broadcast frames from Section 2.

4.8. End State

For a new node, the end state of the joining process is:

* it is synchronized to the network
* it is using the link-layer keying material it learned through the secure joining process
* it has selected one neighbor as its routing parent

* it has one AutoRxCell
* it has one negotiated Tx cell to the selected parent
* it starts to send DIOs, potentially serving as a router for other nodes' traffic
* it starts to send EBs, potentially serving as a JP for new pledges

5. Rules for Adding/Deleting Cells

Once a node has joined the 6TiSCH network, it adds/deletes/relocates cells with the selected parent for three reasons:

* to match the link-layer resources to the traffic between the node and the selected parent (Section 5.1)
* to handle switching parent or(Section 5.2)
* to handle a schedule collision (Section 5.3)

Those cells are called 'negotiated cells' as they are scheduled through 6P, negotiated with the node's parent. Without specific declaration, all cells mentioned in this section are negotiated cells and they are installed at Slotframe 2.

5.1. Adapting to Traffic

A node implementing MSF MUST implement the behavior described in this section.

The goal of MSF is to manage the communication schedule in the 6TiSCH schedule in a distributed manner. For a node, this translates into monitoring the current usage of the cells it has to one of its neighbors, in most cases to the selected parent.

* If the node determines that the number of link-layer frames it is attempting to exchange with the selected parent per unit of time is larger than the capacity offered by the TSCH negotiated cells it has scheduled with it, the node issues a 6P ADD command to that parent to add cells to the TSCH schedule.
* If the traffic is lower than the capacity, the node issues a 6P DELETE command to that parent to delete cells from the TSCH schedule.

The node MUST maintain two separate pairs of the following counters for the selected parent, one for the negotiated Tx cells to that parent and one for the negotiated Rx cells to that parent.

NumCellsElapsed : Counts the number of negotiated cells that have

elapsed since the counter was initialized. This counter is initialized at 0. When the current cell is declared as a negotiated cell to the selected parent, NumCellsElapsed is incremented by exactly 1, regardless of whether the cell is used to transmit/receive a frame.

NumCellsUsed: Counts the number of negotiated cells that have been used. This counter is initialized at 0. NumCellsUsed is incremented by exactly 1 when, during a negotiated cell to the selected parent, either of the following happens:

* The node sends a frame to the parent. The counter increments regardless of whether a link-layer acknowledgment was received or not.
* The node receives a valid frame from the parent. The counter increments only when the frame is a valid IEEE802.15.4 frame.

The cell option of cells listed in CellList in 6P Request frame SHOULD be either (Tx=1, Rx=0) only or (Tx=0, Rx=1) only. Both NumCellsElapsed and NumCellsUsed counters can be used for both type of negotiated cells.

As there is no negotiated Rx Cell installed at initial time, the AutoRxCell is taken into account as well for downstream traffic adaptation. In this case:

* NumCellsElapsed is incremented by exactly 1 when the current cell is AutoRxCell.
* NumCellsUsed is incremented by exactly 1 when the node receives a frame from the selected parent on AutoRxCell.

Implementors MAY choose to create the same counters for each neighbor, and add them as additional statistics in the neighbor table.

The counters are used as follows:

1. Both NumCellsElapsed and NumCellsUsed are initialized to 0 when the node boots.
2. When the value of NumCellsElapsed reaches MAX_NUM_CELLS:
    * If NumCellsUsed > LIM_NUMCELLSUSED_HIGH, trigger 6P to add a single cell to the selected parent
    * If NumCellsUsed < LIM_NUMCELLSUSED_LOW, trigger 6P to remove a single cell to the selected parent
    * Reset both NumCellsElapsed and NumCellsUsed to 0 and go to step 2.

The value of MAX_NUM_CELLS is chosen according to the traffic type of the network. Generally speaking, the larger the value MAX_NUM_CELLS is, the more accurate the cell usage is calculated. The 6P traffic

overhead using a larger value of MAX_NUM_CELLS could be reduced as well. Meanwhile, the latency won't increase much by using a larger value of MAX_NUM_CELLS for periodic traffic type. For bursty traffic, larger value of MAX_NUM_CELLS indeed introduces higher latency. The latency caused by slight changes of traffic load can be absolved by the additional scheduled cells. In this sense, MSF is a scheduling function trading latency with energy by scheduling more cells than needed. Setting MAX_NUM_CELLS to a value at least 4x of the recent maximum number of cells used in a slot frame is RECOMMENDED. For example, a 2 packets/slotframe traffic load results an average 4 cells scheduled (2 cells are used), using at least the value of double number of scheduled cells (which is 8) as MAX_NUM_CELLS gives a good resolution on cell usage calculation.

In case that a node booted or disappeared from the network, the cell reserved at the selected parent may be kept in the schedule forever. A clean-up mechanism MUST be provided to resolve this issue. The clean-up mechanism is implementation-specific. The goal is to confirm those negotiated cells are not used anymore by the associated neighbors and remove them from the schedule.

5.2. Switching Parent

A node implementing MSF SHOULD implement the behavior described in this section.

Part of its normal operation, the RPL routing protocol can have a node switch parent. The procedure for switching from the old parent to the new parent is:

1. the node counts the number of negotiated cells it has per slotframe to the old parent
2. the node triggers one or more 6P ADD commands to schedule the same number of negotiated cells with same cell options to the new parent
3. when that successfully completes, the node issues a 6P CLEAR command to its old parent

For what type of negotiated cell should be installed first, it depends on which traffic has the higher priority, upstream or downstream, which is application-specific and out-of-scope of MSF.

5.3. Handling Schedule Collisions

A node implementing MSF SHOULD implement the behavior described in this section. Other schedule collisions handling algorithm can be an alternative of the algorithm proposed in this section.

Since scheduling is entirely distributed, there is a non-zero probability that two pairs of nearby neighbor nodes schedule a negotiated cell at the same [slotOffset,channelOffset] location in the TSCH schedule. In that case, data exchanged by the two pairs may collide on that cell. We call this case a "schedule collision".

The node MUST maintain the following counters for each negotiated Tx cell to the selected parent:

NumTx: Counts the number of transmission attempts on that cell. Each time the node attempts to transmit a frame on that cell, NumTx is incremented by exactly 1.

NumTxAck: Counts the number of successful transmission attempts on that cell. Each time the node receives an acknowledgment for a transmission attempt, NumTxAck is incremented by exactly 1.

Since both NumTx and NumTxAck are initialized to 0, we necessarily have NumTxAck <= NumTx. We call Packet Delivery Ratio (PDR) the ratio NumTxAck/NumTx; and represent it as a percentage. A cell with PDR=50% means that half of the frames transmitted are not acknowledged.

Each time the node switches parent (or during the join process when the node selects a parent for the first time), both NumTx and NumTxAck MUST be reset to 0. They increment over time, as the schedule is executed and the node sends frames to that parent. When NumTx reaches MAX_NUMTX, both NumTx and NumTxAck MUST be divided by 2. MAX_NUMTX needs to be a power of two to avoid division error. For example, when MAX_NUMTX is set to 256, from NumTx=255 and NumTxAck=127, the counters become NumTx=128 and NumTxAck=64 if one frame is sent to the parent with an Acknowledgment received. This operation does not change the value of the PDR, but allows the counters to keep incrementing. The value of MAX_NUMTX is implementation-specific.

The key for detecting a schedule collision is that, if a node has several cells to the selected parent, all cells should exhibit the same PDR. A cell which exhibits a PDR significantly lower than the others indicates than there are collisions on that cell.

Every HOUSEKEEPINGCOLLISION_PERIOD, the node executes the following steps:

1. It computes, for each negotiated Tx cell with the parent (not for the autonomous cell), that cell’s PDR.

2. Any cell that hasn't yet had NumTx divided by 2 since it was last reset is skipped in steps 3 and 4. This avoids triggering cell relocation when the values of NumTx and NumTxAck are not statistically significant yet.
3. It identifies the cell with the highest PDR.
4. For any other cell, it compares its PDR against that of the cell with the highest PDR. If the subtraction difference between the PDR of the cell and the highest PDR is larger than RELOCATE_PDRTHRES, it triggers the relocation of that cell using a 6P RELOCATE command.

The RELOCATION for negotiated Rx cells is not supported by MSF.

6. 6P SIGNAL command

The 6P SIGNAL command is not used by MSF.

7. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of MSF is IANA_6TISCH_SFID_MSF. How the value of IANA_6TISCH_SFID_MSF is chosen is described in Section 17.

8. Rules for CellList

MSF uses 2-step 6P Transactions exclusively. 6P transactions are only initiated by a node towards its parent. As a result, the cells to put in the CellList of a 6P ADD command, and in the candidate CellList of a RELOCATE command, are chosen by the node initiating the 6P transaction. In both cases, the same rules apply:

* The CellList is RECOMMENDED to have 5 or more cells.
* Each cell in the CellList MUST have a different slotOffset value.
* For each cell in the CellList, the node MUST NOT have any scheduled cell on the same slotOffset.
* The slotOffset value of any cell in the CellList MUST NOT be the same as the slotOffset of the minimal cell (slotOffset=0).
* The slotOffset of a cell in the CellList SHOULD be randomly and uniformly chosen among all the slotOffset values that satisfy the restrictions above.
* The channelOffset of a cell in the CellList SHOULD be randomly and uniformly chosen in [0..numFrequencies], where numFrequencies represents the number of frequencies a node can communicate on.

As a consequence of random cell selection, there is a non-zero chance that nodes in the vicinity installed cells with same slotOffset and channelOffset. An implementer MAY implement a strategy to monitor the candidate cells before adding them in CellList to avoid

collision. For example, a node MAY maintain a candidate cell pool for the CellList. The candidate cells in the pool are pre-configured as Rx cells to promiscuously listen to detect transmissions on those cells. If IEEE802.15.4 transmissions are observed on one cell over multiple iterations of the schedule, that cell is probably used by a TSCH neighbor. It is moved out from the pool and a new cell is selected as a candidate cell. The cells in CellList are picked from the candidate pool directly when required.

9. 6P Timeout Value

The timeout value is calculated for the worst case that a 6P response is received, which means the 6P response is sent out successfully at the very latest retransmission. And for each retransmission, it backs-off with largest value. Hence the 6P timeout value is calculated as ((2^MAXBE)-1)*MAXRETRIES*SLOTFRAME_LENGTH, where:

* MAXBE, defined in IEEE802.15.4, is the maximum backoff exponent used
* MAXRETRIES, defined in IEEE802.15.4, is the maximum retransmission times
* SLOTFRAME_LENGTH represents the length of slotframe

10. Rule for Ordering Cells

Cells are ordered slotOffset first, channelOffset second.

The following sequence is correctly ordered (each element represents the [slottOffset,channelOffset] of a cell in the schedule):

[1,3],[1,4],[2,0],[5,3],[6,0],[6,3],[7,9]

11. Meaning of the Metadata Field

The Metadata field is not used by MSF.

12. 6P Error Handling

Section 6.2.4 of [RFC8480] lists the 6P Return Codes. Figure 1 lists the same error codes, and the behavior a node implementing MSF SHOULD follow.

Code	RECOMMENDED behavior
RC_SUCCESS	nothing
RC_EOL	nothing
RC_ERR	quarantine
RC_RESET	quarantine
RC_ERR_VERSION	quarantine
RC_ERR_SFID	quarantine
RC_ERR_SEQNUM	clear
RC_ERR_CELLLIST	clear
RC_ERR_BUSY	waitretry
RC_ERR_LOCKED	waitretry

Figure 1: Recommended behavior for each 6P Error Code.

The meaning of each behavior from Figure 1 is:

nothing: Indicates that this Return Code is not an error. No error handling behavior is triggered.

clear: Abort the 6P Transaction. Issue a 6P CLEAR command to that neighbor (this command may fail at the link layer). Remove all cells scheduled with that neighbor from the local schedule.

quarantine: Same behavior as for "clear". In addition, remove the node from the neighbor and routing tables. Place the node's identifier in a quarantine list for QUARANTINE_DURATION. When in quarantine, drop all frames received from that node.

waitretry: Abort the 6P Transaction. Wait for a duration randomly and uniformly chosen in [WAIT_DURATION_MIN,WAIT_DURATION_MAX]. Retry the same transaction.

13. Schedule Inconsistency Handling

The behavior when schedule inconsistency is detected is explained in Figure 1, for 6P Return Code RC_ERR_SEQNUM.

14. MSF Constants

Figure 2 lists MSF Constants and their RECOMMENDED values.

Name	RECOMMENDED value
SLOTFRAME_LENGTH	101 slots
NUM_CH_OFFSET	16
MAX_NUM_CELLS	100
LIM_NUMCELLSUSED_HIGH	75
LIM_NUMCELLSUSED_LOW	25
MAX_NUMTX	256
HOUSEKEEPINGCOLLISION_PERIOD	1 min
RELOCATE_PDRTHRES	50 %
QUARANTINE_DURATION	5 min
WAIT_DURATION_MIN	30 s
WAIT_DURATION_MAX	60 s

Figure 2: MSF Constants and their RECOMMENDED values.

15. MSF Statistics

Figure 3 lists MSF Statistics and their RECOMMENDED width.

Name	RECOMMENDED width
NumCellsElapsed	1 byte
NumCellsUsed	1 byte
NumTx	1 byte
NumTxAck	1 byte

Figure 3: MSF Statistics and their RECOMMENDED width.

16. Security Considerations

MSF defines a series of "rules" for the node to follow. It triggers several actions, that are carried out by the protocols defined in the following specifications: the Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration [RFC8180], the 6TiSCH Operation Sublayer Protocol (6P) [RFC8480], and the Constrained Join Protocol (CoJP) for 6TiSCH [I-D.ietf-6tisch-minimal-security]. Confidentiality and authentication of MSF control and data traffic are provided by these specifications whose security considerations continue to apply to MSF. In particular, MSF does not define a new protocol or packet format.

MSF uses autonomous cells for initial bootstrap and the transport of join traffic. Autonomous cells are computed as a hash of nodes' EUI64 addresses. This makes the coordinates of autonomous cell an easy target for an attacker, as EUI64 addresses are visible on the wire and are not encrypted by the link-layer security mechanism. With the coordinates of autonomous cells available, the attacker can launch a selective jamming attack against any nodes' AutoRxCell. If the attacker targets a node acting as a JP, it can prevent pledges from using that JP to join the network. The pledge detects such a situation through the absence of a link-layer acknowledgment for its Join Request. As it is expected that each pledge will have more than one JP available to join the network, one available countermeasure for the pledge is to pseudo-randomly select a new JP when the link to the previous JP appears bad. Such strategy alleviates the issue of the attacker randomly jamming to disturb the network but does not help in case the attacker is targeting a particular pledge. In that case, the attacker can jam the AutoRxCell of the pledge, in order to prevent it from receiving the join response. This situation should be detected through the absence of a particular node from the network and handled by the network administrator through out-of-band means.

MSF adapts to traffic containing packets from the IP layer. It is possible that the IP packet has a non-zero DSCP (Diffserv Code Point [RFC2474]) value in its IPv6 header. The decision how to handle that packet belongs to the upper layer and is out of scope of MSF. As long as the decision is made to hand over to MAC layer to transmit, MSF will take that packet into account when adapting to traffic.

Note that non-zero DSCP value may imply that the traffic is originated at unauthenticated pledges, referring to [I-D.ietf-6tisch-minimal-security]. The implementation at IPv6 layer SHOULD rate-limit this join traffic before it is passed to 6top sublayer where MSF can observe it. In case there is no rate limit for join traffic, intermediate nodes in the 6TiSCH network may be prone to a resource exhaustion attack, with the attacker injecting unauthenticated traffic from the network edge. The assumption is that the rate limiting function is aware of the available bandwidth in the 6top L3 bundle(s) towards a next hop, not directly from MSF, but from an interaction with the 6top sublayer that manages ultimately the bundles under MSF's guidance. How this rate-limit is implemented is out of scope of MSF.

17. IANA Considerations

17.1. MSF Scheduling Function Identifiers

This document adds the following number to the "6P Scheduling Function Identifiers" sub-registry, part of the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry, as defined by [RFC8480]:

SFID	Name	Reference
IANA_6TISCH_SFID_MSF	Minimal Scheduling Function (MSF)	RFC_THIS

Figure 4: New SFID in 6P Scheduling Function Identifiers subregistry.

IANA_6TISCH_SFID_MSF is chosen from range 0-127, which is used for IETF Review or IESG Approval.

18. Contributors

* Beshr Al Nahas (Chalmers University, beshr@chalmers.se)
* Olaf Landsiedel (Chalmers University, olafl@chalmers.se)
* Yasuyuki Tanaka (Inria-Paris, yasuyuki.tanaka@inria.fr)

19. References

19.1. Normative References

[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>.

[RFC8480] Wang, Q., Ed., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)", RFC 8480, DOI 10.17487/RFC8480, November 2018, <https://www.rfc-editor.org/info/rfc8480>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[I-D.ietf-6tisch-minimal-security] Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", Work in Progress, Internet-Draft, draft-ietf-6tisch-minimal-security-15, December 10, 2019, <https://tools.ietf.org/html/draft-ietf-6tisch-minimal-security-15>.

[I-D.ietf-6tisch-enrollment-enhanced-beacon] Dujovne, D. and M. Richardson, "IEEE 802.15.4 Information Element encapsulation of 6TiSCH Join and Enrollment Information", Work in Progress, Internet-Draft, draft-ietf-6tisch-enrollment-enhanced-beacon-14, February 21, 2020, <https://tools.ietf.org/html/draft-ietf-6tisch-enrollment-enhanced-beacon-14>.

[I-D.ietf-6tisch-architecture] Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", Work in Progress, Internet-Draft, draft-ietf-6tisch-architecture-28, October 29, 2019, <https://tools.ietf.org/html/draft-ietf-6tisch-architecture-28>.

[IEEE802154] IEEE standard for Information Technology, "IEEE Std 802.15.4 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", DOI 10.1109/IEEE P802.15.4-REVd/D01, <http://ieeexplore.ieee.org/document/7460875/>.

[SAX-DASFAA]
Ramakrishna, M.V. and J. Zobel, "Performance in Practice of String Hashing Functions", DASFAA , DOI 10.1142/9789812819536_0023, 1997, <https://doi.org/10.1142/9789812819536_0023>.

19.2. 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, <https://www.rfc-editor.org/info/rfc7554>.

[I-D.ietf-6tisch-dtsecurity-zerotouch-join]
Richardson, M., "6tisch Zero-Touch Secure Join protocol", Work in Progress, Internet-Draft, draft-ietf-6tisch-dtsecurity-zerotouch-join-04, July 8, 2019, <https://tools.ietf.org/html/draft-ietf-6tisch-dtsecurity-zerotouch-join-04>.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <https://www.rfc-editor.org/info/rfc6206>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

Appendix A. Example of Implementation of SAX hash function

Considering the interoperability, this section provides an example of implemention SAX hash function [SAX-DASFAA]. The input parameters of the function are:

* T, which is the hashing table length
* c, which is the characters of string s, to be hashed

In MSF, the T is replaced by the length of slotframe 1. String s is replaced by the mote EUI64 address. The characters of the string c0, c1, ..., c7 are the 8 bytes of EUI64 address.

The SAX hash function requires shift operation which is defined as follow:

* L_shift(v,b), which refers to left shift variable v by b bits

* R_shift(v,b), which refers to right shift variable v by b bits

The steps to calculate the hash value of SAX hash function are:

1. initialize variable h to h0 and variable i to 0, where h is the intermediate hash value and i is the index of the bytes of EUI64 address
2. sum the value of L_shift(h,l_bit), R_shift(h,r_bit) and ci
3. calculate the result of exclusive or between the sum value in Step 2 and h
4. modulo the result of Step 3 by T
5. assign the result of Step 4 to h
6. increase i by 1
7. repeat Step2 to Step 6 until i reaches to 8

The value of variable h is the hash value of SAX hash function.

The values of h0, l_bit and r_bit in Step 1 and 2 are configured as:

* h0 = 0
* l_bit = 0
* r_bit = 1

The appropriate values of l_bit and r_bit could vary depending on the the set of motes' EUI64 address. How to find those values is out of the scope of this specification.

Authors' Addresses

Tengfei Chang (editor)
Inria
2 rue Simone Iff
75012 Paris
France

Email: tengfei.chang@inria.fr

Malisa Vucinic
Inria
2 rue Simone Iff
75012 Paris
France

Email: malisa.vucinic@inria.fr

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

Email: xvilajosana@uoc.edu

Simon Duquennoy
RISE SICS
Isafjordsgatan 22
164 29 Kista
Sweden

Email: simon.duquennoy@gmail.com

Diego Dujovne
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

6TiSCH Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2019

M. Tiloca
RISE AB
S. Duquennoy
Yanzi Networks AB
G. Dini
University of Pisa

June 10, 2019

Robust Scheduling against Selective Jamming in 6TiSCH Networks
draft-tiloca-6tisch-robust-scheduling-02

Abstract

This document defines a method to generate robust TSCH schedules in a 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4-2015) network, so as to protect network nodes against selective jamming attack. Network nodes independently compute the new schedule at each slotframe, by altering the one originally available from 6top or alternative protocols, while preserving a consistent and collision-free communication pattern. This method can be added on top of the minimal security framework for 6TiSCH.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 12, 2019.

Table of Contents

1. Introduction

Nodes in a 6TiSCH network communicate using the IEEE 802.15.4-2015 standard and its Timeslotted Channel Hopping (TSCH) mode. Some properties of TSCH make schedule units, i.e. cells, and their usage predictable, even if security services are used at the MAC layer.

This allows an external adversary to easily derive the communication pattern of a victim node. After that, the adversary can perform a selective jamming attack, by covertly, efficiently, and effectively transmitting over the only exact cell(s) in the victim's schedule. For example, this enables the adversary to jeopardize a competitor's network, while still permitting their own network to operate correctly.

This document describes a method to counteract such an attack. At each slotframe, every node autonomously computes a TSCH schedule, as a pseudo-random permutation of the one originally available from 6top [RFC8480] or alternative protocols.

The resulting schedule is provided to TSCH and used to communicate during the next slotframe. In particular, the new communication pattern results unpredictable for an external adversary. Besides, since all nodes compute the same pseudo-random permutation, the new communication pattern remains consistent and collision-free.

The proposed solution is intended to operate on slotframes that are used for data transmission by current network nodes, and that are not used to join the network. In fact, since the TSCH schedule is altered at each slotframe, the proposed method cannot be applied to slotframes that include a "minimal cell" [RFC8180] and possible other randez-vouz cells used for joining the 6TiSCH network.

This document specifies also how this method can be added on top of the minimal security framework for 6TiSCH and its Constrained Join Protocol (CoJP) [I-D.ietf-6tisch-minimal-security].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with terms and concepts defined in [I-D.ietf-6tisch-minimal-security], [I-D.ietf-6tisch-terminology] and [RFC8152].

This document refers also to the following terminology.

- Permutation key. A cryptographic key shared by network nodes and used to permute schedules. Different keys are used to permute the utilization pattern of timeslots and of channelOffsets.

2. Properties of TSCH that Simplify Selective Jamming

This section highlights a number of properties of the TSCH cell usage that greatly simplify the performance of the selective jamming attack described in Section 3.

Given:

- N_S as the size of slotframes in timeslots;
- N_C as the number of available channelOffsets;
- The channel 'f' to communicate at timeslot 's' with ASN and channelOffset 'chOff' computed as f = F[(ASN + chOff) mod N_C];

And assuming for simplicity that:

- N_S and N_C are coprime values;
- The channel hopping sequence is N_C in size and equal to {0, 1, ..., N_C - 1};

Then, the following properties hold:

- Periodicity property. The sequence of channels used for communication by a certain cell repeats with period (N_C x N_S) timeslots.
- Usage property. Within a period, every cell uses all the available channels, each of which only once.
- Offset property. All cells follow the same sequence of channels with a certain offset.
- Predictability property. For each cell, the sequence of channels is predictable. That is, by knowing the channel used by a cell in a given timeslot, it is possible to compute the remaining channel hopping sub-sequence.

  In fact, given a cell active on channel 'f' and timeslot 's' on slotframe 'T', and since ASN = (s + T x N_S), it holds that

  f = [(s + T x N_S + chOff) mod N_C] (Equation 1)

  By solving this equation in 'chOff', one can predict the channels used by the cell in the next sloframes. Note that, in order to do that, one does not need to know the absolute number 'T' of the slotframe (and thus the exact ASN) in which timeslot 's' uses a

certain channel 'f'. In fact, one can re-number slotframes starting from any arbitrarily assumed "starting-slotframe".

3. Selective Jamming Attack

This section describes how an adversary can exploit the properties listed in Section 2, and determine the full schedule of a victim node, even if security services at the MAC layer are used.

This allows the adversary to selectively jam only the exact cell(s) in the victim's schedule, while greatly limiting the exposure to detection. At the same time, the attack is highly effective in jeopardizing victim's communications, and is highly energy-efficient, i.e., can be carried out on battery.

For simplicity, the following description also assumes that a victim node actually transmits/receives during all its allocated cells at each slotframe.

3.1. Adversary Model

This specification addresses an adversary with the following properties.

- The adversary is external, i.e. it does not control any node registered in the 6TiSCH network.
- The adversary wants to target precise network nodes and their traffic. That is, it does not target the 6TiSCH network as a whole, and does not perform a wide-band constant jamming.
- The adversary is able to target multiple victim nodes at the same time. This may require multiple jamming sources and/or multiple antennas per jamming source to carry out the attack.

Furthermore, compared to wide-band constant jamming, the considered selective jamming attack deserves special attention to be addressed, due to the following reasons.

- It is much more energy efficient.
- It minimizes the adversary's exposure and hence the chances to be detected.
- It has the same effectiveness on the intended victim nodes. That is, it achieves the same goal, while avoiding the unnecessarily exposure and costs of wide-band constant jamming.

It is worth noting that a wide-band constant jamming can achieve the same result more easily, in the extreme cases where the target slotframe is (nearly) fully used by a few nodes only, or the adversary has as many antennas as the number of available channels. However, this would still come at the cost of high exposure and higher energy consumption for the adversary.

3.2. Attack Example

The following example considers Figure 1, where N_S = 3, N_C = 4, and the channel hopping sequence is {0,1,2,3}. The shown schedule refers to a network node that uses three cells 'L_1', 'L_2' and 'L_3', with {0,3}, {1,1} and {2,0} as pairs {timeslot, channelOffset}, respectively.

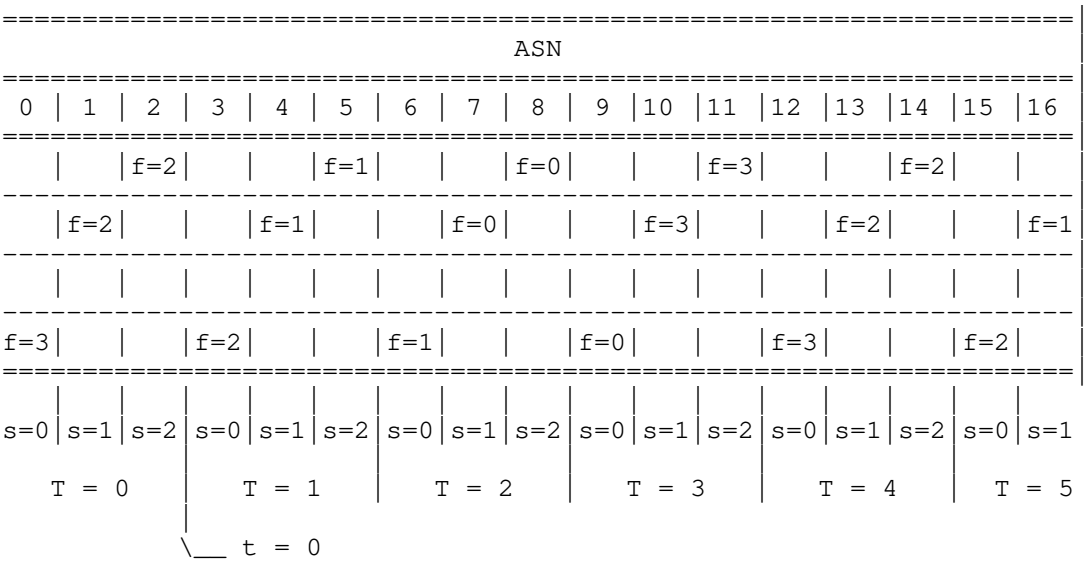

Figure 1: Attack Example with Slotframe Re-numbering

1. The adversary starts the attack at absolute slotframe T = 1, which is assumed as "starting-slotframe" and thus renamed as slotframe t = 0. The renaming is possible due to the offset and predictability properties.

2. The adversary picks a channel 'f*' at random, and monitors it for N_C consecutive slotframes to determine the timeslots in which the victim node communicates on that channel. Due to the usage property, the number of such timeslots is equal to the number of cells assigned to the victim node.

With reference to Figure 1, if, for example, f* = 1, the adversary determines that the victim node uses channel 'f*' in timeslots s = 1 and s = 2 of slotframe t = 0 and in timeslot s = 0 of slotframe t = 1. The adversary can then deduce that the victim node uses three different cells 'L_1', 'L_2' and 'L_3', in timeslots 0, 1 and 2, respectively.

3. The adversary determines the channels on which the victim node is going to transmit in the next slotframes, by exploiting the predictability property.

   That is, by instantiating Equation 1 for cell L_1, timeslot s = 0 and slotframe t = 1, one gets [1 = (3 + chOff_1) mod 4], which has solution for chOff_1 = 2. Hence, the function to predict the channel 'f_1' to be used by cell 'L_1' in a slotframe 't', t >= 1, is f_1 = [(2 + 3 x t) mod 4], which produces the correct periodic sequence of channels {1, 0, 3, 2}. Similarly, one can instantiate Equation 1 for cells 'L_2' and 'L_3', so producing the respective periodic sequence of channels {1,0,3,2} and {1,0,3,2}.

4. The adversary has discovered the full schedule of the victim node and can proceed with the actual selective jamming attack. That is, according to the found schedule, the adversary transmits over the exact cells used by the victim node for transmission/ reception, while staying quiet and saving energy otherwise. This results in a highly effective, highly efficient and hard to detect attack against communications of network nodes.

4. Building Robust Schedules

This section defines a method to protect network nodes against the selective jamming attack described in Section 3. The proposed method alters the communication pattern of all network nodes at every slotframe, in a way unpredictable for the adversary.

At each slotframe 'T', network nodes autonomously compute the communication pattern for the next slotframe 'T+1' as a pseudo-random permutation of the one originally available. In order to ensure that the new communication pattern remains consistent and collision-free, all nodes compute the same permutation of the original one. In particular, at every slotframe, each node separately and independently permutes its timeslot utilization pattern (optionally) as well as its channelOffset utilization pattern.

To perform the required permutations, all network nodes rely on a same secure pseudo-random number generator (SPRNG) as shown in Figure 2, where E(x,y) denotes a cipher which encrypts a plaintext

'y' by means of a key 'x'. Network nodes MUST support the AES-CCM-16-64-128 algorithm.

```
unsigned random(unsigned K, unsigned z) {
    unsigned val = E(K,z);
    return val;
}
```

Figure 2: Secure Pseudo-Random Number Generator

All network nodes share the same following pieces of information.

- K_s, a permutation key used to permute the timeslot utilization pattern, and used as input to the random() function in Figure 2. K_s is provided upon joining the network, and MAY be provided as described in Section 5.

- K_c, a permutation key used to permute the channelOffset utilization pattern, and used as input to the random() function in Figure 2. K_c is provided upon joining the network, and MAY be provided as described in Section 5.

- z_s, a counter used to permute the timeslot utilization pattern, and used as input to the random() function in Figure 2. At the beginning of each slotframe, z_s is equal to [(N_S - 1) x floor(ASN* / N_S)], where ASN* is the ASN value of the first timeslot of that slotframe. Then, z_s grows by (N_S - 1) from the beginning of a slotframe to the beginning of the next one.

- z_c, a counter used to permute the channelOffset utilization pattern, and used as input to the random() function in Figure 2. At the beginning of each slotframe, z_c is equal to [(N_C - 1) x floor(ASN* / N_S)], where ASN* is the ASN value of the first timeslot of that slotframe. Then, z_c grows by (N_C - 1) from the beginning of a slotframe to the beginning of the next one.

Then, at every slotframe, each network node takes the following steps, and generates its own permuted communication schedule to be used at the following slotframe. The actual permutation of cells relies on the well-known Fisher-Yates algorithm, that requires to generate (n - 1) pseudo-random numbers in order to pseudo-randomly shuffle a vector of n elements.

1. First, a pseudo-random permutation is performed on the timeslot dimension of the slotframe. This requires (N_S - 1) invocations of random(K,z), consistently with the Fisher-Yates algorithm. In particular, K = K_s, while z_s is passed as second argument and is incremented by 1 after each invocation. The result of this

step is a permuted timeslot utilization pattern, while the channelOffset utilization pattern is not permuted yet.

2. Second, a pseudo-random permutation is performed on the channelOffset dimension of the slotframe. This requires (N_C - 1) invocations of random(K,z), consistently with the Fisher-Yates algorithm. In particular, K = K_c, while z_c is passed as second argument and is incremented by 1 after each invocation. The result of this step is a fully shuffled communication pattern.

The resulting schedule is then provided to TSCH and considered for sending/receiving traffic during the next slotframe.

As further discussed in Section 6.3, it is possible to skip step 1 above, and hence permute only the channelOffset utilization pattern, while keeping a static timeslot utilization pattern.

Note for implementation: the process described above can be practically implemented by using two vectors, i.e. one for shuffling the timeslot utilization pattern and one for shuffling the channelOffset utilization pattern.

5. Adaptation to the 6TiSCH Minimal Security Framework

The security mechanism described in this specification can be added on top of the minimal security framework for 6TiSCH [I-D.ietf-6tisch-minimal-security].

That is, the two permutation keys K_s and K_c can be provided to a pledge when performing the Constrained Join Protocol (CoJP) defined in Section 8 of [I-D.ietf-6tisch-minimal-security].

To this end, the Configuration CBOR object [RFC7049] used as payload of the Join Response Message and defined in Section 8.4.2 of [I-D.ietf-6tisch-minimal-security] is extended with two new CoJP parameters defined in this specification, namely 'permutation key set' and 'permutation cipher'. The resulting payload of the Join Response message is as follows.

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

The parameter 'permutation key set' is an array encompassing one or two permutation keys encoded as byte strings. That is, the encoding of each individual permutation key is as follows.

```
Permutation_Key = (
      key_value          : bstr
(
```

If the 6TiSCH network uses the security mechanism described in this specification, the parameter 'permutation key set' MUST be included in the CoJP Join Response message and the pledge MUST interpret it as follows.

- In case only one permutation key is present, it is used as K_c to permute the channelOffset utilization pattern, as per Section 4.

- In case two permutation keys are present, the first one is used as K_s to permute the timeslot utilization pattern, while the second one is used as K_c to permute the channelOffset utilization pattern, as per Section 4. The two keys MUST have the same length.

The parameter 'permutation cipher' indicates the encryption algorithm used for the secure pseudo-random number generator as per Figure 2 in Section 4. The value is one of the encryption algorithms defined for COSE [RFC8152], and is taken from Tables 9, 10 and 11 of [RFC8152]. In case the parameter is omitted, the default value of AES-CCM-16-64-128 (COSE algorithm encoding: 10) MUST be assumed.

5.1. Error Handling

In case 'permutation key set' includes two permutation keys with different length or more than two permutation keys, the pledge considers 'permutation key set' not valid and MUST signal the error as specified in Section 8.3.1 of [I-D.ietf-6tisch-minimal-security].

The pledge MUST validate that keys included in 'permutation key set' are appropriate for the encryption algorithm specified in 'permutation cipher' or assumed as default. In case of failed validation, the pledge MUST signal the error as specified in Section 8.3.1 of [I-D.ietf-6tisch-minimal-security].

6. Security Considerations

With reference to Section 3.9 of [RFC7554], this specification achieves an additional "Secure Communication" objective, namely it defines a mechanism to build and enforce a TSCH schedule which is

robust against selective jamming attack, while at the same time consistent and collision-free.

Furthermore, the same security considerations from the minimal security framework for 6TiSCH [I-D.ietf-6tisch-minimal-security] hold for this document. The rest of this section discusses a number of additional security considerations.

6.1. Effectiveness of Schedule Shuffling

The countermeasure defined in Section 4 practically makes each node's schedule look random to an external observer. Hence, it prevents the adversary from performing the attack described in Section 3.

Then, a still available strategy for the adversary is to jam a number of cells selected at random, possibly on a per-slotframe basis. This considerably reduces the attack effectiveness in successfully jeopardizing victims' communications.

At the same time, nodes using different cells than the intended victims' would experience an overall slightly higher fraction of corrupted messages. In fact, the communications of such accidental victims might be corrupted by the adversary, when they occur during a jammed timeslot and exactly over the channelOffset chosen at random.

6.2. Renewal of Key Material

It is RECOMMENDED that the two permutation keys K_s and K_c are revoked and renewed every time a node leaves the network. This prevents a leaving node to keep the permutation keys, which may be exploited to selectively jam communications in the network.

This rekeying operation is supposed to be performed anyway upon every change of network membership, in order to preserve backward and forward security. In particular, new IEEE 802.15.4 link-layer keys are expected to be distributed before a new pledge can join the network, or after one or more nodes have left the network.

The specific approach to renew the two permutation keys, possibly together with other security material, is out of the scope of this specification.

6.3. Static Timeslot Allocations

As mentioned in Section 4 and Section 5, it is possible to permute only the channelOffset utilization pattern, while preserving the originally scheduled timeslot utilization pattern. This can be desirable, or even unavoidable in some scenarios, in order to

guarantee end-to-end latencies in multi-hop networks, as per accordingly designed schedules.

However, preserving a static timeslot utilization pattern would considerably increase the attack surface for a random jammer adversary. That is, the adversary would immediately learn the timeslot utilization pattern of a victim node, and would have a chance to successfully jam a victim's cell equal to (1 / N_C).

6.4. Network Joining Through Randez-vous Cells

As described in [I-D.ietf-6tisch-minimal-security], a pledge joins a 6TiSCH network through a Join Proxy (JP), according to the Constrained Join Protocol (CoJP) and based on the information conveyed in broadcast Enhanced Beacons (EBs). In particular, the pledge will communicate with the JP over randez-vous cells indicated in the EBs.

In practice, such cells are commonly part of a separate slotframe, which includes one scheduled "minimal cell" [RFC8180], typically used also for broadcasting EBs. Such slotframe, i.e. Slotframe 0, usually differs from the slotframe(s) used for both EBs and data transmission.

In order to keep the join process feasible and deterministic, the solution described in this specification is not applied to Slotframe 0 or any other slotframes that include randez-vous cells for joining. As a consequence, an adversary remains able to selectively jam the "minimal cell" (or any randez-vous cell used for joining), so potentially jeopardizing the CoJP and preventing pledges to join the network altogether.

7. IANA Considerations

This document has the following actions for IANA.

7.1. Permutation Key Set

IANA is asked to enter the following value into the "Constrained Join Protocol Parameters Registry" defined in [I-D.ietf-6tisch-minimal-security] and within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TISCH) parameters" registry.

Name	Label	CBOR type	Description	Reference
permutation key set	TBD	array	Identifies the array including one or two permutation keys to alter cell utilization	[[this document]]

7.2. Permutation Cipher

IANA is asked to enter the following value into the "Constrained Join Protocol Parameters Registry" defined in [I-D.ietf-6tisch-minimal-security] and within the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TISCH) parameters" registry.

Name	Label	CBOR type	Description	Reference
permutation cipher	TBD	integer	Identifies the cipher used for generating pseudo-random numbers to alter cell utilization	[[this document]]

8. References

8.1. Normative References

[I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-10 (work in progress), April 2019.

[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>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[I-D.ietf-6tisch-terminology] Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terms Used in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-10 (work in progress), March 2018.

[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>.

[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>.

[RFC8480] Wang, Q., Ed., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)", RFC 8480, DOI 10.17487/RFC8480, November 2018, <https://www.rfc-editor.org/info/rfc8480>.

[Test-Implementation] "Test Implementation in C with OpenSSL", May 2019, <https://gitlab.com/crimson84/draft-tiloca-6tisch-robust-scheduling/tree/master/test>.

Appendix A. Test Vector

This appendix provides a test vector for an example where the method proposed in this document is used to generate robust TSCH schedules.

The example focuses on one network node and considers the schedule in Figure 1 as the original schedule to permute at each slotframe.

The results shown in this example have been produced using the implementation available at [Test-Implementation].

A.1. Detailed Technique

In this example, the permutation of the timeslot utilization pattern and of the channelOffset utilization pattern occurs as follows.

A.1.1. Data Structures and Schedule Encoding

Each network node maintains two vectors X_s and X_c, each composed of N_S unsigned integer values. At the beginning of each slotframe, X_s and X_c indicate the node's original schedule. In particular:

- X_s indicates the usage of timeslots in the slotframe. That is, the element X_s[i] refers to the i-th timeslot of the slotframe.
- X_c indicates the usage of channelOffsets at each timeslot in the slotframe. That is, X_c[i] refers to the channelOffset value used in the i-th timeslot of the slotframe.

Then, the two vectors encode the schedule information as follows:

- If the i-th timeslot is not used, X_s[i] = 0 and X_c[i] = N_C.
- If the i-th timeslot is used to transmit with channelOffset 'c', X_s[i] = 1 and X_c[i] = c.
- If the i-th timeslot is used to receive with channelOffset 'c', X_s[i] = 2 and X_c[i] = c.

Note that optimized implementations can achieve the same goal with permutation vectors of smaller size.

A.1.2. Pseudo-Random Number Generation

When invoking E() within the random() function in Figure 2:

- The second parameter has 5 bytes in size like the ASN, and is provided as plaintext to the permutation cipher.
- A copy of the second parameter is left-padded with 8 octects with value 0x00. The result is provided as 13-byte nonce to the permutation cipher, i.e. AES-CCM-16-64-128 in this example.
- No additional authenticated data are provided to the permutation cipher.

The unsigned value returned by E() and random() is the computed ciphertext left-padded with 3 octects with value 0x00. That is, the returned value is 8 bytes in size.

A.1.3. Array Permutation

To produce the required permutations, this example considers the Fisher-Yates modern version in Figure 3, which requires (n - 1) swaps to shuffle an array of n elements.

```
// Shuffle an array 'a' of 'n' elements (indices 0, ... , n - 1)
for i from (n - 1) down to 1 do {
   j = random integer such that 0 <= j <= i;
   exchange a[j] and a[i];
}
```

Figure 3: Fisher-Yates algorithm

At each step of the loop, 'j' is computed as r % (i + 1), where '%' is the modulo operator, and 'r' is the value returned by the function random() in Figure 2, as described in Appendix A.1.2.

A.1.4. Schedule Permutation

At each slotframe, the original schedule is considered as starting point to produce the permuted schedule for the following slotframe.

In particular, the permuted schedule for the following slotframe is computed according to the following steps.

1. The same pseudo-random permutation is performed on both vectors X_s and X_c, by using the Fisher-Yates algorithm in Figure 3. This requires (N_S - 1) invocations of random(K,z). In particular, K = K_s, while z_s is passed as second argument and is incremented by 1 after each invocation. As a result, X_s specifies the permuted timeslot utilization pattern, whereas X_c specifies a consistent while temporary channelOffset utilization pattern.

2. A vector Y of size N_C is produced, as a permutation of {0, 1, ..., N_C - 1} performed by using the Fisher-Yates algorithm in Figure 3. This requires (N_C - 1) invocations of random(K,z). In particular, K = K_c, while z_c is passed as second argument and is incremented by 1 after each invocation.

3. The vector X_c is updated as follows. Each element X_c[i] that refers to a non active timeslot, i.e. X_c[i] = N_C, is left as is. Otherwise, X_c[i] takes as value Y[j], where j = X_c[i].

As a result, the two permuted vectors X_s and X_c together provide a full communication pattern to use during the next slotframe.

A.2. Test Configuration

```
N_S = 3 // Slotframe size, in timeslots

N_C = 4 // Available channel offsets

Channel hopping sequence = {0, 1, 2, 3}

X_s = {1, 1, 2} // Original timeslot utilization pattern {Tx, Tx, Rx}

X_c = {3, 1, 0} // Original channelOffset utilization pattern

Starting ASN = 0

Permutation cipher: AES-CCM-16-64-128

K_s = { 0xce, 0xb0, 0x09, 0xae, 0xa4, 0x45, 0x44, 0x51,
        0xfe, 0xad, 0xf0, 0xe6, 0xb3, 0x6f, 0x45, 0x55 }

K_c = { 0xce, 0xb0, 0x09, 0xae, 0xa4, 0x45, 0x44, 0x51,
        0xfe, 0xad, 0xf0, 0xe6, 0xb3, 0x6f, 0x45, 0x56 }
```

A.3. Example Output

```
****************** ****************** ******************

START ROUND 1 of 2

The slotframe starts with: ASN = 0; z_s = 0; z_c = 0

****************** ****************** ******************

-- Start shuffling the time offsets --

---------- ---------- ----------

Counter (z_s): 0

Plaintext: 0x0000000000 (5 bytes)

Cipher nonce: 0x00000000000000000000000000 (13 bytes)

Ciphertext: 0xbedca72db3 (5 bytes)

Padded ciphertext: 0x000000bedca72db3 (8 bytes)

Fisher-Yates swap index i: 2
```

```
   Fisher-Yates swap-index j: 0
   ---------- ---------- ----------
   Counter (z_s): 1
   Plaintext: 0x0000000001 (5 bytes)
   Cipher nonce: 0x00000000000000000000000001 (13 bytes)
   Ciphertext: 0x23d36801f1 (5 bytes)
   Padded ciphertext: 0x00000023d36801f1 (8 bytes)
   Fisher-Yates swap index i: 1
   Fisher-Yates swap-index j: 1
   ---------- ---------- ----------
   -- Intermediate schedule --
   Timeslot utilization pattern X_s = {2, 1, 1}
   ChannelOffset utilization pattern X_c = {0, 1, 3}
   ---------- ---------- ----------
   -- Start shuffling the channel offset schedule --
   ---------- ---------- ----------
   Counter (z_c): 0
   Plaintext: 0x0000000000 (5 bytes)
   Cipher nonce: 0x00000000000000000000000000 (13 bytes)
   Ciphertext: 0x1e957fe44d (5 bytes)
   Padded ciphertext: 0x0000001e957fe44d (8 bytes)
   Fisher-Yates swap index i: 3
   Fisher-Yates swap-index j: 1
   ---------- ---------- ----------
```

Counter (z_c): 1

Plaintext: 0x0000000001 (5 bytes)

Cipher nonce: 0x00000000000000000000000001 (13 bytes)

Ciphertext: 0x6e2b990263 (5 bytes)

Padded ciphertext in bytes: 0x0000006e2b990263 (8 bytes)

Fisher-Yates swap index i: 2

Fisher-Yates swap-index j: 2

---------- ---------- ----------

Counter (z_c): 2

Plaintext: 0x0000000002 (5 bytes)

Cipher nonce: 0x00000000000000000000000002 (13 bytes)

Ciphertext: 0x4fae2cfe22 (5 bytes)

Padded ciphertext: 0x0000004fae2cfe22 (8 bytes)

Fisher-Yates swap index i: 1

Fisher-Yates swap-index j: 0

---------- ---------- ----------

Next slotframe starting with ASN = 3 will use:

- Shuffled timeslot schedule {2, 1, 1}, i.e. {Rx, Tx, Tx}.
- Shuffled channel offset schedule {3, 0, 1}.
- Shuffled frequencies schedule {2, 0, 2}.

********** ********** **********

START ROUND 2 OF 2

The slotframe starts with: ASN = 3; z_s = 2; z_c = 3

********** ********** **********

```
   -- Start shuffling the time offsets --
   ---------- ---------- ----------
   Counter (z_s): 2
   Plaintext: 0x0000000002 (5 bytes)
   Cipher nonce: 0x00000000000000000000000002 (13 bytes)
   Ciphertext: 0xd9a0c0f8eb (5 bytes)
   Padded ciphertext: 0x000000d9a0c0f8eb (8 bytes)
   Fisher-Yates swap index i: 2
   Fisher-Yates swap-index j: 2
   ---------- ---------- ----------
   Counter (z_s): 3
   Plaintext: 0x0000000003 (5 bytes)
   Cipher nonce: 0x00000000000000000000000003 (13 bytes)
   Ciphertext: 0x7aabd818ac (5 bytes)
   Padded ciphertext: 0x0000007aabd818ac (8 bytes)
   Fisher-Yates swap index i: 1
   Fisher-Yates swap-index j: 0
   ---------- ---------- ----------
   -- Intermediate schedules --
   Timeslot utilization pattern X_s = {1, 1, 2}
   ChannelOffset utilization pattern X_c = {1, 3, 0}
   ---------- ---------- ----------
   -- Start shuffling the channel offset schedule --
   ---------- ---------- ----------
```

Counter (z_c): 3

Plaintext: 0x0000000003 (5 bytes)

Cipher nonce: 0x00000000000000000000000003 (13 bytes)

Ciphertext: 0x947cf7c1d4 (5 bytes)

Padded ciphertext: 0x000000947cf7c1d4 (8 bytes)

Fisher-Yates swap index i: 3

Fisher-Yates swap-index j: 0

---------- ---------- ----------

Counter (z_c): 4

Plaintext: 0x0000000004 (5 bytes)

Cipher nonce: 0x00000000000000000000000004 (13 bytes)

Ciphertext: 0xa9255744e7 (5 bytes)

Padded ciphertext: 0x000000a9255744e7 (8 bytes)

Fisher-Yates swap index i: 2

Fisher-Yates swap-index j: 1

---------- ---------- ----------

Counter (z_c): 5

Plaintext: 0x0000000005 (5 bytes)

Cipher nonce: 0x00000000000000000000000005 (13 bytes)

Ciphertext: 0xa70a456e9e (5 bytes)

Padded ciphertext: 0x000000a70a456e9e (8 bytes)

Fisher-Yates swap index i: 1

Fisher-Yates swap-index j: 0

---------- ---------- ----------

Next slotframe starting with ASN = 6 will use:

- Shuffled timeslot schedule {1, 1, 2}, i.e. {Tx, Tx, Rx}.
- Shuffled channel offset schedule {3, 0, 2}.
- Shuffled frequencies schedule {1, 3, 2}.

Acknowledgments

The authors sincerely thank Tengfei Chang, Michael Richardson, Yasuyuki Tanaka, Pascal Thubert and Malisa Vucinic for their comments and feedback.

The work on this document has been partly supported by the EIT-Digital High Impact Initiative ACTIVE, and by the VINNOVA and Celtic-Next project CRITISEC.

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: marco.tiloca@ri.se

Simon Duquennoy
Yanzi Networks AB
Isafjordsgatan 32C
Kista SE-16440 Stockholm
Sweden

Email: simon.duquennoy@yanzinetworks.com

Gianluca Dini
University of Pisa
Largo L. Lazzarino 2
Pisa 56122
Italy

Email: gianluca.dini@unipi.it