

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

T. Chang, Ed.
Inria
M. Vucinic
University of Montenegro
X. Vilajosana
Universitat Oberta de Catalunya
March 1, 2018

6TiSCH Minimal Scheduling Function (MSF)
draft-chang-6tisch-msf-01

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 builds upon the 6top 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 [RFC2119].

Status of This Memo

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

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

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

This Internet-Draft will expire on September 2, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction	3
2.	Interface to the Minimal 6TiSCH Configuration	4
3.	Node Behavior at Boot	5
3.1.	Start State	5
3.2.	Step 1 - Choosing Frequency	5
3.3.	Step 2 - Receiving EBs	5
3.4.	Step 3 - Join Request/Response	6
3.5.	Step 4 - Acquiring a RPL rank	6
3.6.	Step 5 - 6P ADD to Preferred Parent	6
3.7.	Step 6 - Send EBs and DIOs	7
3.8.	Step 7 - Neighbor Polling	7
3.9.	End State	7
4.	Rules for Adding/Deleting Cells	7
4.1.	Adapting to Traffic	8
4.2.	Switching Parent	9
4.3.	Handling Schedule Collisions	9
5.	6P SIGNAL command	11
6.	Scheduling Function Identifier	11
7.	Rules for CellList	11
8.	6P Timeout Value	11
9.	Rule for Ordering Cells	11
10.	Meaning of the Metadata Field	12
11.	6P Error Handling	12
12.	Schedule Inconsistency Handling	12
13.	MSF Constants	13
14.	MSF Statistics	13
15.	Security Considerations	13
16.	IANA Considerations	14
16.1.	MSF Scheduling Function Identifiers	14
17.	References	14
17.1.	Normative References	14

17.2. Informative References	15
Appendix A. Implementation Status	15
Appendix B. Performance Evaluation	16
Appendix C. [TEMPORARY] Changelog	16
Authors' Addresses	16

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 [I-D.ietf-6tisch-6top-protocol]: it complements [I-D.ietf-6tisch-6top-protocol] by providing the rules of when to add/delete cells in the communication schedule. The SF defined in this document follows that definition, and satisfies all the requirements for an SF listed in Section 4.2 of [I-D.ietf-6tisch-6top-protocol].

MSF builds on top of the following specifications: the Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration [RFC8180], the 6top Protocol (6P) [I-D.ietf-6tisch-6top-protocol], 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 7 steps described in Section 3. The end state of the join process is that the node is synchronized to the network, has mutually authenticated to the network, has identified a preferred routing parent, and has scheduled a single dedicated cell to that parent. After the join process, the node can continuously add/delete/relocate cells, as described in Section 4. It does so for 3 reasons: to match the link-layer resources to the traffic, to handle changing parent, to handle a schedule collision.

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 root). Appendix B contains a performance evaluation of MSF.

This specification follows the recommended structure of an SF specification in Appendix A of [I-D.ietf-6tisch-6top-protocol], with the following adaptations:

- o We have reordered part of the sections, in particular to have the section on the node behavior at boot Section 3 appear early in this specification.

- o We added sections on the interface to the minimal 6TiSCH configuration Section 2, the use of the SIGNAL command Section 5, the MSF constants Section 13, the MSF statistics Section 14, the performance of MSF Appendix B.
- o We have not included an examples section.

2. Interface to the Minimal 6TiSCH Configuration

A node implementing MSF MUST implement the Minimal 6TiSCH Configuration [RFC8180], which defines the "minimal cell", a single shared cell providing minimal connectivity between the nodes in the network.

MSF uses the minimal cell to exchange the following packets:

1. Enhanced Beacons (EBs), defined by [IEEE802154-2015]. These are broadcast frames.
2. DODAG Information Objects (DIOs), defined by [RFC6550]. These are broadcast frames.
3. The Join Request and Join Response packets sent between the pledge and the Join Proxy, defined by [I-D.ietf-6tisch-minimal-security]. These are unicast frames.
4. 6P packets to schedule the first dedicated cell with a neighbor. These are unicast frames.

Because the minimal cell is SHARED, the back-off algorithm defined in [IEEE802154-2015] is used to resolve collisions. To ensure there is enough bandwidth available on the minimal cell for the unicast traffic, a node implementing MSF SHOULD enforce the following rules for broadcast frames:

1. send EBs on a portion of the minimal cells not exceeding $1/(3(N+1))$, where N is the number of neighbors of the node.
2. send DIOs on a portion of the minimal cells not exceeding $1/(3(N+1))$, where N is the number of neighbors of the node.

The RECOMMENDED behavior for sending EBs is to have a node send EBs with a probability of $1/(3(N+1))$. The RECOMMENDED behavior for sending DIOs is to use a Trickle timer with rate-limiting.

Section 3.3 describes how to evaluate the number of neighbors during the joining process. After the joining process, how to evaluate the number of neighbors is implementation-specific.

As detailed in Section 2.2 of [I-D.ietf-6tisch-6top-protocol], MSF MUST schedule cells from Slotframe 1, while Slotframe 0 is used for traffic defined in the Minimal 6TiSCH Configuration. The length of Slotframe 0 and Slotframe 1 MUST be the same value. The default of

SLOTFRAME_LENGTH is RECOMMENDED, although any value can be advertised in the EBs.

3. 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. Section 3.1 details the start state; Section 3.9 details the end state. The other sections detail the 7 steps of the joining process. We use the term "pledge" and "joined node", as defined in [I-D.ietf-6tisch-minimal-security].

3.1. Start State

A node implementing MSF MUST implement the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security]. As a corollary, this means that a pledge, before being switched on, is pre-configured with the Pre-Shared Key (PSK) for joining, as well as any other configuration detailed in [I-D.ietf-6tisch-minimal-security].

3.2. Step 1 - Choosing Frequency

When switched on, the pledge SHOULD randomly choose a frequency among the available frequencies, and start listening for EBs on that frequency.

3.3. Step 2 - Receiving EBs

Upon receiving the first EB, the pledge SHOULD continue 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

While the exact behavior is implementation-specific, the RECOMMENDED behavior is to follow [RFC8180], and listen until EBs sent by NUM_NEIGHBOURS_TO_WAIT nodes (defined in [RFC8180]) have been received.

During this step, the pledge MAY synchronize to any EB it receives 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.richardson-6tisch-join-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].

3.4. Step 3 - Join Request/Response

As per [I-D.ietf-6tisch-minimal-security], after having selected a JP, the pledge sends a Join Request to its JP. Because no dedicated cells are in place at this point, this happens on the minimal cell. The JP then forwards the Join Request to the JRC, possibly over multiple hops. When forwarding this Join Request, a node MUST use a dedicated cell it has with its preferred parent (not the minimal cell). How such dedicated cells are installed is detailed in Section 3.6.

As per [I-D.ietf-6tisch-minimal-security], the JRC sends back a Join Response to the pledge, through the JP. When forwarding this Join Response, a node MUST use a dedicated cell it has with its child (not the minimal cell). How such dedicated cells are installed is detailed in Section 3.6.

As per [I-D.ietf-6tisch-minimal-security], after receiving the Join Response, the pledge learns the keying material used in the network, as well as other configurations, and becomes a "joined node".

3.5. Step 4 - Acquiring a RPL rank

Because it has learned the link-layer keying material used in the network, the joined node can now decrypt the DIO packets sent by its neighbors. Per [RFC6550], the joined node receives DIOs, computes its own rank, and selects a preferred parent.

3.6. Step 5 - 6P ADD to Preferred Parent

After having selected a preferred parent, the joined node MUST issue a 6P ADD command to that parent, with the following fields:

- o CellOptions: set to TX=1,RX=1,SHARED=1
- o NumCells: set to 1
- o CellList: at least 5 cells, chosen according to Section 7

After the 6P Transaction is finished, the node MUST synchronize only to its preferred parent. At this point, the node has a dedicated cell which allows for bidirectional communication with its preferred parent.

3.7. Step 6 - Send EBs and DIOs

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

3.8. Step 7 - Neighbor Polling

The node SHOULD send some form of keep-alive messages to all its neighbors it has dedicated cells with. The Keep-Alive (KA) mechanism is detailed in [RFC7554]. It uses the keep-alive messages to its preferred parent to stay synchronized. It uses the keep-alive messages to its children (with which it has a dedicated cell to, which the child has created) to ensure the child is still reachable. The RECOMMENDED period for sending keep-alive messages is KA_PERIOD.

If the keep-alive message to a child fails at the link layer (i.e. the maximum number of link-layer retries is reached), the node SHOULD declare the child as unreachable. This can happen for example when the child node is switched off.

When a neighbor is declared unreachable, the node MUST remove all dedicated cells with that neighbor from its own schedule. In addition, it MAY issue a 6P CLEAR to that neighbor (which can fail at the link-layer).

3.9. End State

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

- o it is synchronized to the network
- o it is using the link-layer keying material it learned through the secure joining process
- o it has identified its preferred routing parent
- o it has a single dedicated cell to its preferred routing parent
- o it is periodically sending DIOs, potentially serving as a router for other nodes' traffic
- o it is periodically sending EBs, potentially serving as a JP for new joining nodes

4. Rules for Adding/Deleting Cells

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

- o to match the link-layer resources to the traffic between the node and its preferred parent (Section 4.1)
- o to handle switching preferred parent (Section 4.2)
- o to handle a schedule collision (Section 4.3)

4.1. 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 its preferred parent:

- o If the node determines that the number of link-layer frames it is attempting to exchange with its preferred parent per unit of time is larger than the capacity offered by the TSCH cells it has scheduled with it, it triggers a 6P Transaction with its preferred parent to add cells to the TSCH schedule of both nodes.
- o If the traffic is lower than the capacity, the node triggers a 6P Transaction with its preferred parent to delete cells from the TSCH schedule of both nodes.

From the join process, the node already has one dedicated cell scheduled to its preferred parent. A node MUST NOT remove all cells to its preferred parent, i.e. there must always be at least one dedicated cell scheduled between a node and its preferred parent, even if no frames are being exchanged between them.

Adding/removing/relocating cells involves exchanging frames that contain 6P commands. Once the first cell has been established as part of the join process, all 6P frames MUST be sent on the dedicated cells (not the minimal cell).

The node MUST maintain the following counters for its preferred parent:

NumCellsPassed: Counts the number of dedicated cells that have passed since the counter was initialized. This counter is initialized at 0. Each time the TSCH state machine indicates the current cell is a dedicated cell to the preferred parent, NumCellsPassed is incremented by exactly 1, regardless of whether the cell is used to transmit/receive a frame.

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

- * The node sends a frame to its preferred parent. The counter increments regardless of whether a link-layer acknowledgment was received or not.
- * The node receives a frame from its preferred parent.

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 NumCellsPassed and NumCellsUsed are initialized to 0 when the node boots.
2. When the value of NumCellsPassed reaches MAX_NUMCELLS:
 - * If NumCellsUsed > LIM_NUMCELLSUSED_HIGH, trigger 6P to add a single cell to the preferred parent
 - * If NumCellsUsed < LIM_NUMCELLSUSED_LOW, trigger 6P to remove a single cell to the preferred parent
 - * Reset both NumCellsPassed and NumCellsUsed to 0 and go to step 2.

4.2. Switching Parent

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

Part of its normal operation, the RPL routing protocol can have a node switch preferred parents. The procedure for switching from the old preferred parent to the new preferred parent is:

1. the node counts the number of dedicated cells it has per slotframe to the old preferred parent
2. the node triggers one or more 6P ADD commands to schedule the same number of dedicated cells to the new preferred parent
3. when that successfully completes, the node issues a 6P CLEAR command to its old preferred parent

4.3. Handling Schedule Collisions

A node implementing MSF SHOULD implement the behavior described in this section. The "MUST" statements in this section hence only apply if the node implements schedule collision handling.

Since scheduling is entirely distributed, there is a non-zero probability that two pairs of nearby neighbor nodes schedule a 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 cell to its preferred 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.

Implementors MAY choose to maintain the same counters for each cell in the schedule.

Since both NumTx and NumTxAck are initialized to 0, we necessarily have $\text{NumTxAck} \leq \text{NumTx}$. We call Packet Delivery Ratio (PDR) the ratio $\text{NumTxAck}/\text{NumTx}$; and represent it as a percentage. A cell with PDR=50% means that half of the frames transmitted are not acknowledged (and need to be retransmitted).

Each time the node switches preferred parent (or during the join process when the node selects a preferred 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 its preferred parent. When NumTx reaches 256, both NumTx and NumTxAck MUST be divided by 2. That is, for example, from NumTx=256 and NumTxAck=128, they become NumTx=128 and NumTxAck=64. This operation does not change the value of the PDR, but allows the counters to keep incrementing.

The key for detecting a schedule collision is that, if a node has several cells to the same preferred parent, all cells should exhibit the same PDR. A cell which exhibits a PDR significantly lower than the others indicates that there are collisions on that cell.

Every HOUSEKEEPINGCOLLISION_PERIOD, the node executes the following steps:

1. It computes, for each cell with its preferred parent, 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 each other cell, it compares its PDR against that of the cell with the highest PDR. If it's less than RELOCATE_PDRTHRES, it triggers the relocation of that cell using a 6P RELOCATE command.

5. 6P SIGNAL command

The 6P SIGNAL command is not used by MSF.

6. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of MSF is IANA_6TISCH_SFID_MSF.

7. Rules for CellList

MSF uses 2-step 6P Transactions exclusively. 6P Transactions are only initiated by a node towards its preferred 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:

- o the CellList SHOULD contain 5 or more cells.
- o Each cell in the CellList MUST have a different slotOffset value.
- o For each cell in the CellList, the node MUST NOT have any scheduled cell on the same slotOffset.
- o The slotOffset value of any cell in the CellList MUST NOT be the same as the slotOffset of the minimal cell (slotOffset=0).
- o The slotOffset of a cell in the CellList SHOULD be randomly and uniformly chosen among all the slotOffset values that satisfy the restrictions above.
- o 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.

8. 6P Timeout Value

The 6P Timeout is not a constant value. It is calculated as $(1/C)*(1/PDR)*SIXP_TIMEOUT_SEC_FACTOR$, where:

- o C represents the number of cells per second scheduled to that neighbor
- o PDR represents the average PDR of those cells
- o SIXP_TIMEOUT_SEC_FACTOR is a security factor, a constant

9. Rule for Ordering Cells

Cells are ordered slotOffset first, channelOffset second.

The following sequence is correctly ordered (each element represents the [slotOffset,channelOffset] of a cell in the schedule):

[1,3],[1,4],[2,0],[5,3],[6,0],[6,3],[7,9]

10. Meaning of the Metadata Field

The Metadata field is not used by MSF.

11. 6P Error Handling

Section 6.2.4 of [I-D.ietf-6tisch-6top-protocol] 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 hence 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. Keep that node in the neighbor and routing tables.

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 [WAITDURATION_MIN, WAITDURATION_MAX]. Retry the same transaction.

12. Schedule Inconsistency Handling

The behavior when schedule inconsistency is detected is explained in Figure 1, for 6P Return Code RC_ERR_SEQNUM.

13. MSF Constants

Figure 2 lists MSF Constants and their RECOMMENDED values.

Name	RECOMMENDED value
KA_PERIOD	10 s
LIM_NUMCELLSUSED_HIGH	75 %
LIM_NUMCELLSUSED_LOW	25 %
HOUSEKEEPINGCOLLISION_PERIOD	1 min
RELOCATE_PDRTHRES	50 %
SIXP_TIMEOUT_SEC_FACTOR	3 x
SLOTFRAME_LENGTH	101 slots
QUARANTINE_DURATION	5 min
WAITDURATION_MIN	30 s
WAITDURATION_MAX	60 s

Figure 2: MSF Constants and their RECOMMENDED values.

14. MSF Statistics

Figure 3 lists MSF Statistics and their RECOMMENDED width.

Name	RECOMMENDED width
NumCellsPassed	1 byte
NumCellsUsed	1 byte
NumTx	1 byte
NumTxAck	1 byte

Figure 3: MSF Statistics and their RECOMMENDED width.

15. 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 6top Protocol (6P) [I-D.ietf-6tisch-6top-protocol], and the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security]. In particular, MSF does not define a new protocol or packet format.

MSF relies entirely on the security mechanisms defined in the specifications listed above.

16. IANA Considerations

16.1. 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 [I-D.ietf-6tisch-6top-protocol]:

SFID	Name	Reference
IANA_6TISCH_SFID_MSF	Minimal Scheduling Function (MSF)	RFCXXXX (NOTE:this)

Figure 4: IETF IE Subtype '6P'.

17. References

17.1. Normative References

[I-D.ietf-6tisch-6top-protocol]

Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-09 (work in progress), October 2017.

[I-D.ietf-6tisch-minimal-security]

Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-04 (work in progress), October 2017.

[I-D.richardson-6tisch-join-enhanced-beacon]

Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational Element encapsulation of 6tisch Join Information", draft-richardson-6tisch-join-enhanced-beacon-03 (work in progress), January 2018.

[IEEE802154-2015]

IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015.

[RFC2119]

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

- [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>>.
- [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>>.

17.2. Informative References

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

Appendix A. Implementation Status

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

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of

running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

OpenWSN: MSF is being implemented in the OpenWSN project [OpenWSN] under a BSD open-source license. The authors of this document are collaborating with the OpenWSN community to gather feedback about the status and performance of the protocols described in this document. Results from that discussion will appear in this section in future revision of this specification. More information about this implementation at <http://www.openwsn.org/>.
6TiSCH simulator The 6TiSCH simulator is a Python-based high-level simulator on which MSF is being implemented. More information at <https://bitbucket.org/6tisch/simulator/>.

Appendix B. Performance Evaluation

The performance of MSF may be published as companion documents to this specification, possibly under the form a applicability statements.

Appendix C. [TEMPORARY] Changelog

- o draft-chang-6tisch-msf-01
 - * When neighbor is unreachable, sending a CLEAR command was a MUST, now a MAY.
 - * Fixing 6P Timeout calculation.
 - * Clearer text for "Handling Schedule Collisions" section.
 - * Typos.
 - * Input from Yasuyuki Tanaka's review (<https://www.ietf.org/mail-archive/web/6tisch/current/msg05723.html>).
- o draft-chang-6tisch-msf-00
 - * Initial submission.

Authors' Addresses

Tengfei Chang (editor)
Inria
2 rue Simone Iff
Paris 75012
France

Email: tengfei.chang@inria.fr

Malisa Vucinic
University of Montenegro
Dzordza Vasingtona bb
Podgorica 81000
Montenegro

Email: malisav@ac.me

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

Email: xvilajosana@uoc.edu

6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

T. Chang, Ed.
Inria
M. Vucinic
University of Montenegro
X. Vilajosana
Universitat Oberta de Catalunya
S. Duquennoy
RISE SICS
D. Dujovne, Ed.
Universidad Diego Portales
July 2, 2018

6TiSCH Minimal Scheduling Function (MSF)
draft-chang-6tisch-msf-02

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 builds 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 [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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Interface to the Minimal 6TiSCH Configuration	4
3. Autonomous Unicast Cells	5
4. Node Behavior at Boot	6
4.1. Start State	6
4.2. Step 1 - Choosing Frequency	6
4.3. Step 2 - Receiving EBs	6
4.4. Step 3 - Setting up Autonomous Unicast Cells	7
4.5. Step 4 - Join Request/Response	7
4.6. Step 5 - Acquiring a RPL rank	7
4.7. Step 6 - Send EBs and DIOs	7
4.8. Step 7 - Neighbor Polling	7
4.9. End State	8
5. Rules for Adding/Deleting Cells	8
5.1. Adapting to Traffic	8
5.2. Switching Parent	10
5.3. Handling Schedule Collisions	10
6. 6P SIGNAL command	12
7. Scheduling Function Identifier	12
8. Rules for CellList	12
9. 6P Timeout Value	12
10. Rule for Ordering Cells	12
11. Meaning of the Metadata Field	13
12. 6P Error Handling	13
13. Schedule Inconsistency Handling	13
14. MSF Constants	14
15. MSF Statistics	14
16. Security Considerations	14
17. IANA Considerations	15
17.1. MSF Scheduling Function Identifiers	15
18. References	15

18.1. Normative References	15
18.2. Informative References	16
Appendix A. Contributors	16
Appendix B. Implementation Status	17
Appendix C. Performance Evaluation	17
Appendix D. [TEMPORARY] Changelog	17
Authors' Addresses	18

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 [I-D.ietf-6tisch-6top-protocol]: it complements [I-D.ietf-6tisch-6top-protocol] by providing the rules of when to add/delete cells in the communication schedule. The SF defined in this document follows that definition, and satisfies all the requirements for an SF listed in Section 4.2 of [I-D.ietf-6tisch-6top-protocol].

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) [I-D.ietf-6tisch-6top-protocol], 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 7 steps described in Section 4. The end state of the join process is that the node is synchronized to the network, has mutually authenticated to the network, has identified a preferred routing parent, has scheduled one default unicast cell to/from each of its neighbors. 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, to handle a schedule collision.

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 root). Appendix C contains a performance evaluation of MSF.

This specification follows the recommended structure of an SF specification in Appendix A of [I-D.ietf-6tisch-6top-protocol], with the following adaptations:

- o We have reordered part of the sections, in particular to have the section on the node behavior at boot Section 4 appear early in this specification.
- o 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, the MSF statistics Section 15, the performance of MSF Appendix C.
- o This specification does not include an examples section.

2. Interface to the Minimal 6TiSCH Configuration

A node implementing MSF MUST implement the Minimal 6TiSCH Configuration [RFC8180], which defines the "minimal cell", a single shared cell providing minimal connectivity between the nodes in the network.

MSF uses the minimal cell to exchange the following packets:

1. Enhanced Beacons (EBs), defined by [IEEE802154-2015]. These are broadcast frames.
2. DODAG Information Objects (DIOs), defined by [RFC6550]. These are broadcast frames.

Because the minimal cell is SHARED, the back-off algorithm defined in [IEEE802154-2015] is used to resolve collisions. To ensure there is enough bandwidth available on the minimal cell, a node implementing MSF SHOULD enforce the following rules for broadcast frames:

1. send EBs on a portion of the minimal cells not exceeding $1/(3(N+1))$, where N is the number of neighbors of the node.
2. send DIOs on a portion of the minimal cells not exceeding $1/(3(N+1))$, where N is the number of neighbors of the node.

The RECOMMENDED behavior for sending EBs is to have a node send EBs with a probability of $1/(3(N+1))$. The RECOMMENDED behavior for sending DIOs is to use a Trickle timer with rate-limiting.

Section 4.3 describes how to evaluate the number of neighbors during the joining process. After the joining process, how to evaluate the number of neighbors is implementation-specific.

As detailed in Section 2.2 of [I-D.ietf-6tisch-6top-protocol], MSF MUST schedule cells from Slotframe 1, while Slotframe 0 is used for traffic defined in the Minimal 6TiSCH Configuration. The length of Slotframe 0 and Slotframe 1 SHOULD be the same value. The default of SLOTFRAME_LENGTH is RECOMMENDED, although any value can be advertised in the EBs.

3. Autonomous Unicast Cells

MSF nodes MUST initialize Slotframe 1 with a set of default cells for unicast communication with their neighbors. These cells are referred to as 'autonomous cells', because they are maintained autonomously by each node. Each node has:

1. One cell to receive, at a [slotOffset,channelOffset] computed as a hash of the node's EUI64 (detailed next). The cell options for this cell are RX=1.
2. For each neighbor in the IPv6 neighbor table, one cell to transmit, at a [slotOffset,channelOffset] computed as a hash of the neighbor's EUI64 (detailed next). The cell options for this cell are TX=1, SHARED=1.

To compute a [slotOffset,channelOffset] from an EUI64 address, nodes MUST use the hash function SAX [SAX-DASFAA]. The coordinates are computed to distribute the cells across all 16 channel offsets, and all but the first time offsets 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) % 16
```

Because of hash collisions, there are cases where one node has multiple cells scheduled at the same time offset and/or channel offset. Note that nodes have only one autonomous RX cell and potentially multiple TX cells. Hash collisions among a set of cells at a given time offset is resolved at run-time as follows:

1. The TX cell with the most packets in outgoing queue takes precedence.
2. If all TX cells have empty outgoing queues, the RX cell takes precedence.

Throughout the network lifetime, nodes MUST maintain the autonomous cells as follows:

1. The receive cell MUST always remain scheduled.
2. Whenever a new neighbor is discovered, add a transmit cell for it.
3. Whenever a new neighbor is removed, remove transmit cell that was assigned to it.
4. 6P CLEAR MUST NOT erase autonomous cells.

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. Section 4.1 details the start state; Section 4.9 details the end state. The other sections detail the 7 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 MUST implement the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security]. As a corollary, this means that a pledge, before being switched on, is pre-configured with the Pre-Shared Key (PSK) for joining, as well as any other configuration detailed in [I-D.ietf-6tisch-minimal-security].

4.2. Step 1 - Choosing Frequency

When switched on, the pledge SHOULD randomly choose a frequency among the available frequencies, and start listening for EBs on that frequency.

4.3. Step 2 - Receiving EBs

Upon receiving the first EB, the pledge SHOULD continue 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

While the exact behavior is implementation-specific, the RECOMMENDED behavior is to follow [RFC8180], and listen until EBs sent by NUM_NEIGHBOURS_TO_WAIT nodes (defined in [RFC8180]) have been received.

During this step, the pledge MAY synchronize to any EB it receives 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.richardson-6tisch-join-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 Unicast Cells

After joining, nodes MUST set up their autonomous unicast cells, as described in Section 3. This enables unicast communication in Slotframe 1, until more cells are added with 6P as defined in Section 5.

4.5. Step 4 - Join Request/Response

As per [I-D.ietf-6tisch-minimal-security], after having selected a JP, the pledge sends a Join Request to its JP. Because no dedicated cells are in place at this point, this happens on the autonomous unicast cell. The JP then forwards the Join Request to the JRC, possibly over multiple hops. When forwarding this Join Request, a node MUST use a unicast cell (autonomous or dedicated) it has with its preferred parent. How dedicated cells are installed is detailed in Section 5.

As per [I-D.ietf-6tisch-minimal-security], the JRC sends back a Join Response to the pledge, through the JP. When forwarding this Join Response, a node MUST use a unicast (autonomous or dedicated) cell it has with its child (not the minimal cell).

As per [I-D.ietf-6tisch-minimal-security], after receiving the Join Response, the pledge learns the keying material used in the network, as well as other configurations, and becomes a "joined node".

4.6. Step 5 - Acquiring a RPL rank

Because it has learned the link-layer keying material used in the network, the joined node can now decrypt the DIO packets sent by its neighbors. Per [RFC6550], the joined node receives DIOs, computes its own rank, and selects a preferred parent.

4.7. Step 6 - Send EBs and DIOs

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

4.8. Step 7 - Neighbor Polling

The node SHOULD send some form of keep-alive messages to all its neighbors it has unicast cells with. The Keep-Alive (KA) mechanism is detailed in [RFC7554]. It uses the keep-alive messages to its preferred parent to stay synchronized. It uses the keep-alive messages to its children (with which it has a unicast cell to) to ensure the child is still reachable. The RECOMMENDED period for sending keep-alive messages is KA_PERIOD.

If the keep-alive message to a child fails at the link layer (i.e. the maximum number of link-layer retries is reached), the node SHOULD declare the child as unreachable. This can happen for example when the child node is switched off.

When a neighbor is declared unreachable, the node MUST remove all dedicated cells with that neighbor from its own schedule. In addition, it MAY issue a 6P CLEAR to that neighbor (which can fail at the link-layer). If the node has autonomous cells to the unreachable neighbor those cells will be removed following the procedure described in Section 3.

4.9. End State

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

- o it is synchronized to the network
- o it is using the link-layer keying material it learned through the secure joining process
- o it has identified its preferred routing parent
- o it has a set of autonomous unicast cells to/from its neighbors
- o it is periodically sending DIOs, potentially serving as a router for other nodes' traffic
- o it is periodically sending EBs, potentially serving as a JP for new joining nodes

5. Rules for Adding/Deleting Cells

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

- o to match the link-layer resources to the traffic between the node and its preferred parent (Section 5.1)
- o to handle switching preferred parent (Section 5.2)
- o to handle a schedule collision (Section 5.3)

5.1. Adapting to Traffic

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

In order to handle transient traffic bursts, MSF uses the [IEEE802154-2015] frame pending bit (page 152, Section 7.2.1.3). By setting the bit, a node can transmit a series of packets to a given neighbor in consecutive time offsets. The next paragraphs define how to handle longer-term fluctuations in traffic, using 6P.

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 its preferred parent:

- o If the node determines that the number of link-layer frames it is attempting to exchange with its preferred parent per unit of time is larger than the capacity offered by the TSCH unicast cells (dedicated and autonomous cells) it has scheduled with it, the node triggers a 6P Transaction with its preferred parent to add dedicated cells to the TSCH schedule of both nodes.
- o If the traffic is lower than the capacity, the node triggers a 6P Transaction with its preferred parent to delete dedicated cells from the TSCH schedule of both nodes.

From the join process, the node already has a set of autonomous unicast cells, as defined in Section 3. The autonomous cells **MUST** NOT be removed by 6P, so that there always exists a unicast cell between a node and its preferred parent, even if no frames are being exchanged between them. Autonomous cells are used indistinguishably together with dedicated cells, for broadcast or unicast traffic with the target neighbor. The procedure to remove autonomous cells is described in Section 3.

Adding/removing/relocating cells involves exchanging frames that contain 6P commands. All 6P frames **MUST** be sent on the unicast cells (and not the minimal cell).

The node **MUST** maintain the following counters for its preferred parent:

NumCellsPassed: Counts the number of unicast cells (dedicated and autonomous cells) that have passed since the counter was initialized. This counter is initialized at 0. Each time the TSCH state machine indicates that the current cell is a unicast cell to the preferred parent, NumCellsPassed is incremented by exactly 1, regardless of whether the cell is used to transmit/receive a frame.

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

- * The node sends a frame to its preferred parent. The counter increments regardless of whether a link-layer acknowledgment was received or not.
- * The node receives a frame from its preferred parent.

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 NumCellsPassed and NumCellsUsed are initialized to 0 when the node boots.
2. When the value of NumCellsPassed reaches MAX_NUMCELLS:
 - * If NumCellsUsed > LIM_NUMCELLSUSED_HIGH, trigger 6P to add a single cell to the preferred parent
 - * If NumCellsUsed < LIM_NUMCELLSUSED_LOW, trigger 6P to remove a single cell to the preferred parent
 - * Reset both NumCellsPassed and NumCellsUsed to 0 and go to step 2.

5.2. Switching Parent

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

Part of its normal operation, the RPL routing protocol can have a node switch preferred parents. The procedure for switching from the old preferred parent to the new preferred parent is:

1. the node counts the number of dedicated (unicast but not autonomous) cells it has per slotframe to the old preferred parent
2. the node triggers one or more 6P ADD commands to schedule the same number of dedicated cells to the new preferred parent
3. when that successfully completes, the node issues a 6P CLEAR command to its old preferred parent

5.3. Handling Schedule Collisions

A node implementing MSF SHOULD implement the behavior described in this section. The "MUST" statements in this section hence only apply if the node implements schedule collision handling.

Since scheduling is entirely distributed, there is a non-zero probability that two pairs of nearby neighbor nodes schedule a 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 cell to its preferred 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.

Implementors MAY choose to maintain the same counters for each cell in the schedule.

Since both NumTx and NumTxAck are initialized to 0, we necessarily have $\text{NumTxAck} \leq \text{NumTx}$. We call Packet Delivery Ratio (PDR) the ratio $\text{NumTxAck}/\text{NumTx}$; and represent it as a percentage. A cell with PDR=50% means that half of the frames transmitted are not acknowledged (and need to be retransmitted).

Each time the node switches preferred parent (or during the join process when the node selects a preferred 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 its preferred parent. When NumTx reaches 256, both NumTx and NumTxAck MUST be divided by 2. That is, for example, from NumTx=256 and NumTxAck=128, they become NumTx=128 and NumTxAck=64. This operation does not change the value of the PDR, but allows the counters to keep incrementing.

The key for detecting a schedule collision is that, if a node has several cells to the same preferred parent, all cells should exhibit the same PDR. A cell which exhibits a PDR significantly lower than the others indicates that there are collisions on that cell.

Every HOUSEKEEPINGCOLLISION_PERIOD, the node executes the following steps:

1. It computes, for each dedicated cell with its preferred 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 each other cell, it compares its PDR against that of the cell with the highest PDR. If it's less than RELOCATE_PDRTHRES, it triggers the relocation of that cell using a 6P RELOCATE command.

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.

8. Rules for CellList

MSF uses 2-step 6P Transactions exclusively. 6P Transactions are only initiated by a node towards its preferred 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:

- o The CellList SHOULD contain 5 or more cells.
- o Each cell in the CellList MUST have a different slotOffset value.
- o For each cell in the CellList, the node MUST NOT have any scheduled cell on the same slotOffset.
- o The slotOffset value of any cell in the CellList MUST NOT be the same as the slotOffset of the minimal cell (slotOffset=0).
- o The slotOffset of a cell in the CellList SHOULD be randomly and uniformly chosen among all the slotOffset values that satisfy the restrictions above.
- o 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.

9. 6P Timeout Value

The 6P Timeout is not a constant value. It is calculated as $(1/C)*(1/PDR)*SIXP_TIMEOUT_SEC_FACTOR$, where:

- o C represents the number of cells per second scheduled to that neighbor
- o PDR represents the average PDR of those cells
- o SIXP_TIMEOUT_SEC_FACTOR is a security factor, a constant

10. Rule for Ordering Cells

Cells are ordered slotOffset first, channelOffset second.

The following sequence is correctly ordered (each element represents the [slotOffset,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 [I-D.ietf-6tisch-6top-protocol] 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. Keep that node in the neighbor and routing tables.

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 `[WAITDURATION_MIN, WAITDURATION_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
KA_PERIOD	10 s
LIM_NUMCELLSUSED_HIGH	75 %
LIM_NUMCELLSUSED_LOW	25 %
HOUSEKEEPINGCOLLISION_PERIOD	1 min
RELOCATE_PDRTHRES	50 %
SIXP_TIMEOUT_SEC_FACTOR	3 x
SLOTFRAME_LENGTH	101 slots
QUARANTINE_DURATION	5 min
WAITDURATION_MIN	30 s
WAITDURATION_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
NumCellsPassed	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) [I-D.ietf-6tisch-6top-protocol], and the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security]. In particular, MSF does not define a new protocol or packet format.

MSF relies entirely on the security mechanisms defined in the specifications listed above.

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 [I-D.ietf-6tisch-6top-protocol]:

SFID	Name	Reference
IANA_6TISCH_SFID_MSF	Minimal Scheduling Function (MSF)	RFCXXXX (NOTE: this)

Figure 4: IETF IE Subtype '6P'.

18. References

18.1. Normative References

[I-D.ietf-6tisch-6top-protocol]

Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer Protocol (6P)", draft-ietf-6tisch-6top-protocol-12 (work in progress), June 2018.

[I-D.ietf-6tisch-minimal-security]

Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-06 (work in progress), May 2018.

[I-D.richardson-6tisch-join-enhanced-beacon]

Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational Element encapsulation of 6tisch Join Information", draft-richardson-6tisch-join-enhanced-beacon-03 (work in progress), January 2018.

[IEEE802154-2015]

IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015.

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

18.2. Informative References

- [OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.
- [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>>.
- [SAX-DASFAA] Ramakrishna, M. and J. Zobel, "Performance in Practice of String Hashing Functions", DASFAA , 1997.

Appendix A. Contributors

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

Appendix B. Implementation Status

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

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

OpenWSN: MSF is being implemented in the OpenWSN project [OpenWSN] under a BSD open-source license. The authors of this document are collaborating with the OpenWSN community to gather feedback about the status and performance of the protocols described in this document. Results from that discussion will appear in this section in future revision of this specification. More information about this implementation at <http://www.openwsn.org/>.
6TiSCH simulator The 6TiSCH simulator is a Python-based high-level simulator on which MSF is being implemented. More information at <https://bitbucket.org/6tisch/simulator/>.

Appendix C. Performance Evaluation

The performance of MSF may be published as companion documents to this specification, possibly under the form a applicability statements.

Appendix D. [TEMPORARY] Changelog

- o draft-chang-6tisch-msf-02
 - * Added autonomous cell.
- o draft-chang-6tisch-msf-01
 - * When neighbor is unreachable, sending a CLEAR command was a MUST, now a MAY.

- * Fixing 6P Timeout calculation.
- * Clearer text for "Handling Schedule Collisions" section.
- * Typos.
- * Input from Yasuyuki Tanaka's review (<https://www.ietf.org/mail-archive/web/6tisch/current/msg05723.html>).
- o draft-chang-6tisch-msf-00
 - * Initial submission.

Authors' Addresses

Tengfei Chang (editor)
Inria
2 rue Simone Iff
Paris 75012
France

Email: tengfei.chang@inria.fr

Malisa Vucinic
University of Montenegro
Dzordza Vasingtona bb
Podgorica 81000
Montenegro

Email: malisav@ac.me

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

Email: xvilajosana@uoc.edu

Simon Duquennoy
RISE SICS
Isafjordsgatan 22
164 29 Kista
Sweden

Email: simon.duquennoy@ri.se

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

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

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

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

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

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 2, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

Authors' Addresses 14

1. TEMPORARY EDITORIAL NOTES

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

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

2. Introduction

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

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

2.1. Application Domains

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

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

3. General Operation

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

3.1. Cell Coordinates

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

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

3.2. Types of Slotframes

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

3.2.1. Rendez-vous slotframe

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

3.2.2. Receiver-based slotframe

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

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

3.2.3. Sender-based slotframe

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

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

3.3. Conditional Cells

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

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

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

3.4. Interaction between Slotframes

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

4. Configuration

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

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

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

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

Figure 1 describes the format of the ASF SIGNAL command.

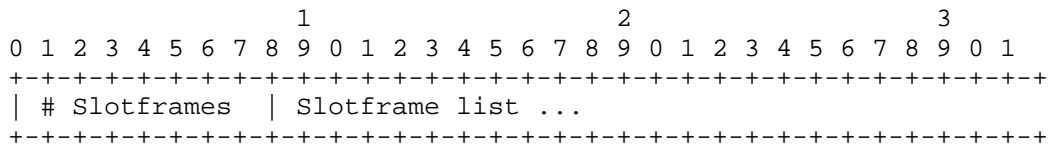


Figure 1: Format of the ASF SIGNAL command.

Where:

- # Slotframes: The number of ASF slotframes
- Slotframe list: The list of slotframes, with format described in Figure 2

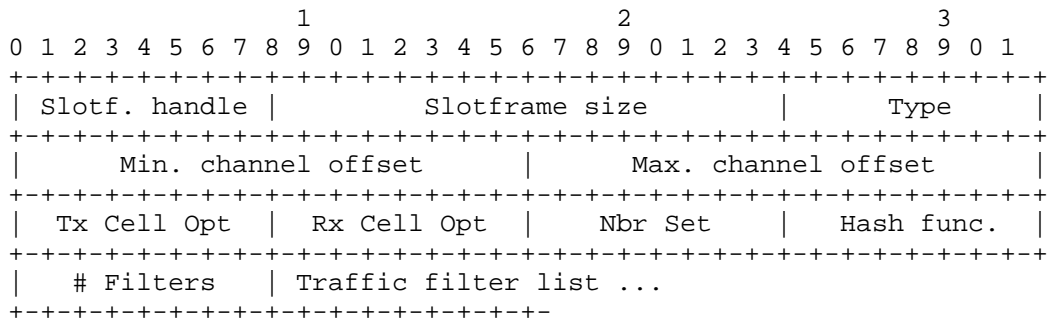


Figure 2: Format of the ASF SIGNAL slotframe descriptor.

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

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

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

Figure 3: Field: types of slotframes.

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

Figure 4: Field: neighbor set.

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

Figure 5: Field: Hash function

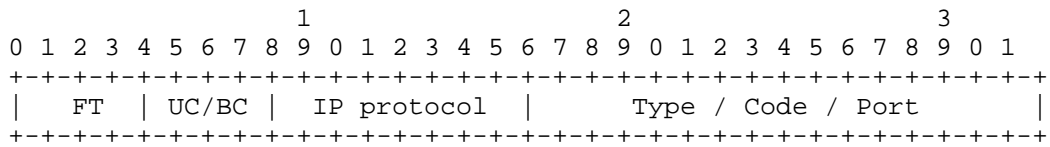


Figure 6: Format of the ASF SIGNAL traffic filters.

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

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

Example filters are given next:

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

5. Scheduling Function Identifier

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

6. Rules for Adding/Deleting Cells

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

7. Rules for CellList

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

8. 6P Timeout Value

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

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

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

9. Rule for Ordering Cells

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

10. Meaning of the Metadata Field

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

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

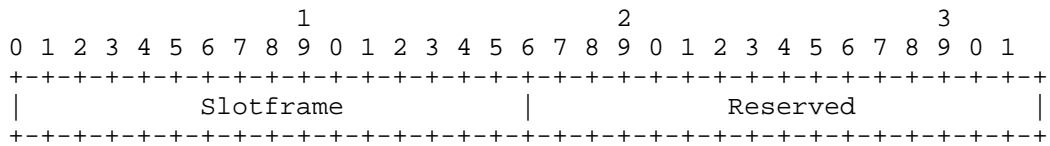


Figure 7: Format of the Metadata Field.

11. Node Behavior at Boot

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

12. 6P Error Handling

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

13. Examples

TODO

14. [TEMPORARY] Implementation Status

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

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

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

15. Security Considerations

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

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

16. IANA Considerations

16.1. 6P Scheduling Function Identifiers 'ASF'

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

SFID	Name	Reference
IANA_6TiSCH_SFID_ASF	Autonomous Scheduling Function (ASF)	RFCXXXX

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

17. References

17.1. Normative References

[IEEE802154-2015] IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015.

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

17.2. Informative References

[Contiki] Dunkels, A., Lignan, A., Thebaudeau, B., Quattlebaum, R., Rosendal, F., Oikonomou, G., Deru, L., Alvira, M., Tsiftes, N., Schmidt, O., and S. Duquennoy, "The Contiki Open Source OS for the Internet of Things", <https://github.com/contiki-os/contiki>, November 2016.

[I-D.ietf-6tisch-6top-protocol] Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-09 (work in progress), October 2017.

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

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

[Orchestra-SenSys]

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

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

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

[SAX-DASFAA]

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

Appendix A. Contributors

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

Appendix B. Acknowledgments

TODO people

TODO projects

Appendix C. [TEMPORARY] Changelog

o draft-duquennoy-6tisch-asf-01

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

o draft-duquennoy-6tisch-asf-00

* Initial draft.

Authors' Addresses

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

Email: simon.duquennoy@ri.se

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

Email: xvilajosana@uoc.edu

Thomas Watteyne
Inria
2 Rue Simone Iff
Paris
France

Email: thomas.watteyne@inria.fr

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

Q. Wang, Ed.
Univ. of Sci. and Tech. Beijing
X. Vilajosana
Universitat Oberta de Catalunya
T. Watteyne
Analog Devices
March 5, 2018

6top Protocol (6P)
draft-ietf-6tisch-6top-protocol-10

Abstract

This document defines the 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 6P 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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction	3
2.	6TiSCH Operation Sublayer (6top)	4
2.1.	Hard/Soft Cells	5
2.2.	Using 6P with the Minimal 6TiSCH Configuration	5
3.	6top Protocol (6P)	6
3.1.	6P Transactions	6
3.1.1.	2-step 6P Transaction	7
3.1.2.	3-step 6P Transaction	9
3.2.	Message Format	10
3.2.1.	6top Information Element (IE)	10
3.2.2.	Generic 6P Message Format	10
3.2.3.	6P CellOptions	11
3.2.4.	6P CellList	13
3.3.	6P Commands and Operations	14
3.3.1.	Adding Cells	14
3.3.2.	Deleting Cells	16
3.3.3.	Relocating Cells	17
3.3.4.	Counting Cells	22
3.3.5.	Listing Cells	23
3.3.6.	Clearing the Schedule	25
3.3.7.	Generic Signaling Between SFs	26
3.4.	Protocol Functional Details	26
3.4.1.	Version Checking	26
3.4.2.	SFID Checking	27
3.4.3.	Concurrent 6P Transactions	27
3.4.4.	6P Timeout	28
3.4.5.	Aborting a 6P Transaction	28
3.4.6.	SeqNum Management	28
3.4.7.	Handling Error Responses	34
3.5.	Security	34
4.	Requirements for 6top Scheduling Functions (SF)	34

- 4.1. SF Identifier (SFID) 34
- 4.2. Requirements for an SF 34
- 5. Security Considerations 35
- 6. IANA Considerations 35
 - 6.1. IETF IE Subtype '6P' 35
 - 6.2. 6TiSCH parameters sub-registries 36
 - 6.2.1. 6P Version Numbers 36
 - 6.2.2. 6P Message Types 36
 - 6.2.3. 6P Command Identifiers 37
 - 6.2.4. 6P Return Codes 38
 - 6.2.5. 6P Scheduling Function Identifiers 39
 - 6.2.6. 6P CellOptions bitmap 40
- 7. References 41
 - 7.1. Normative References 41
 - 7.2. Informative References 41
- Appendix A. Recommended Structure of an SF Specification 42
- Appendix B. Implementation Status 42
- Appendix C. [TEMPORARY] Changelog 43
- Authors' Addresses 47

1. Introduction

All communication in a 6TiSCH network is orchestrated by a schedule [RFC7554]. The schedule is composed of cells, each identified by a [slotOffset,channelOffset]. This specification defines the 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 in each other. This results in distributed schedule management in a 6TiSCH network. The 6top layer terminates the 6top Protocol, and runs one of 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 the requirements for an SF are defined here.

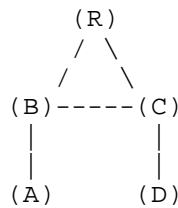


Figure 1: A simple 6TiSCH network.

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

node that issues a 6P request; node B the node that receives this request.

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

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

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

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

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

2. 6TiSCH Operation Sublayer (6top)

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

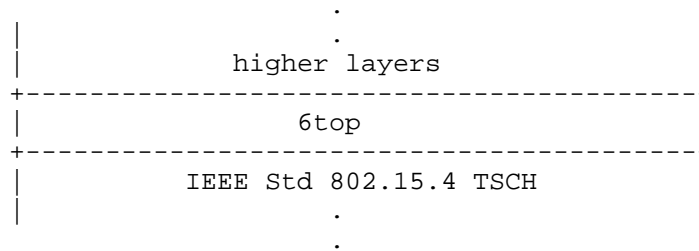


Figure 2: The 6top sublayer in the protocol stack.

The roles of the 6top sublayer are to:

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

2.1. Hard/Soft Cells

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

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

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

2.2. Using 6P with the Minimal 6TiSCH Configuration

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

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

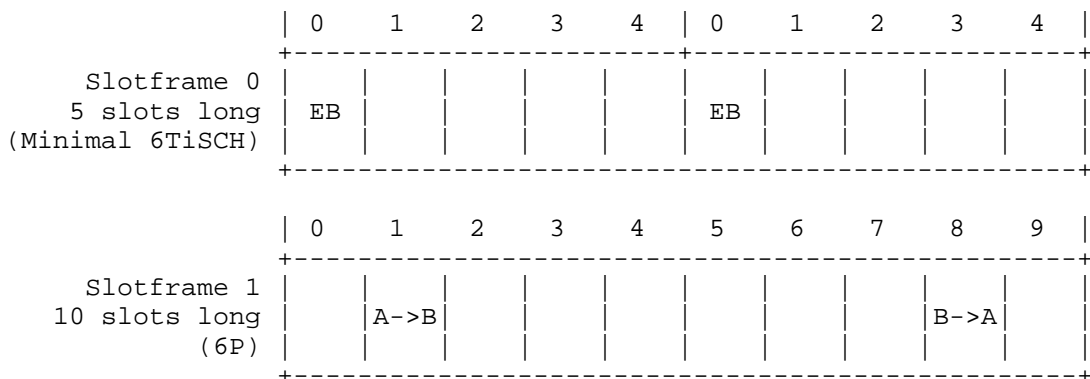


Figure 3: 2-slotframe structure when using 6P alongside the Minimal 6TiSCH Configuration.

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 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 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.

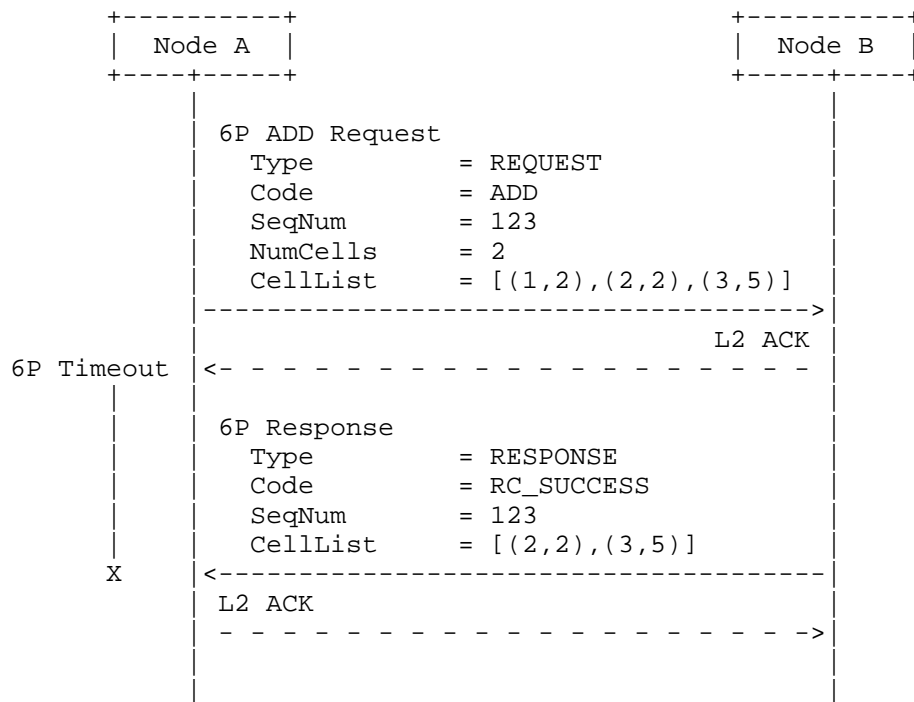


Figure 4: An example 2-step 6P Transaction.

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 3 candidate cells.
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, 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 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.

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.

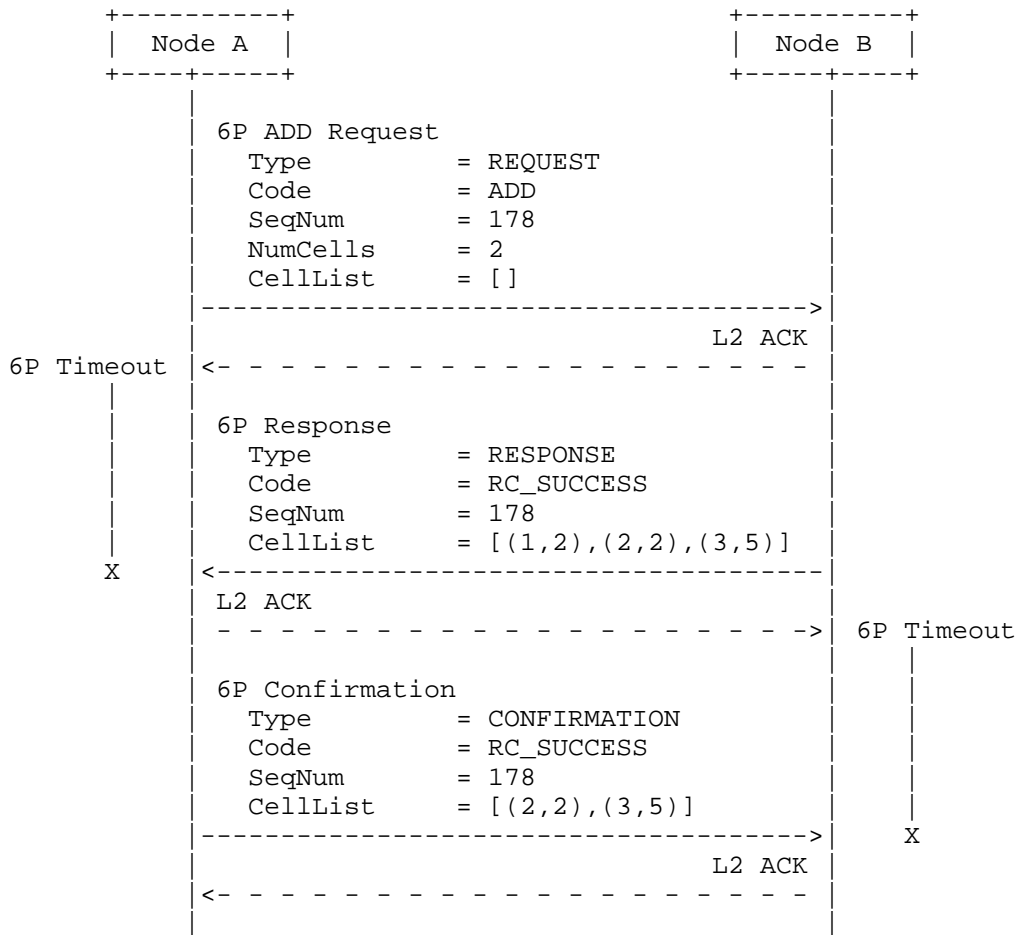


Figure 5: An example 3-step 6P Transaction.

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, but 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.
4. The SF running on node B selects 3 candidate cells. Node B sends back a 6P Response to node A, indicating the 3 cells it 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.
6. The SF running on node A selects 2 cells from the CellList field in the 6P Response. 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 this 6P Transaction, 2 cells from A to B have been added to the TSCH schedule of both nodes A and B.

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 with 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.

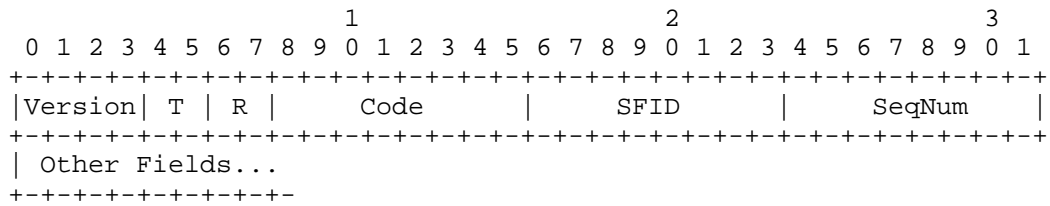


Figure 6: Generic 6P Message Format.

- 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. 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.

3.2.3. 6P CellOptions

An 8-bit 6P CellOptions bitmap is present in the following 6P requests: ADD, DELETE, COUNT, LIST, RELOCATE.

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

- o In the 6P RELOCATE request, the 6P CellOptions bitmap is used to specify what type of cell to relocate.

The contents of the 6P CellOptions bitmap apply to all elements in the CellList field. Section 6.2.6 contains the RECOMMENDED format of the 6P CellOptions bitmap. Figure 7 contains the RECOMMENDED meaning of the 6P CellOptions bitmap for the 6P COUNT and 6P LIST requests. Figure 8 contains the RECOMMENDED 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 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
TX=0,RX=1,S=0	all cells marked as TX
TX=1,RX=1,S=0	all cells marked as TX and RX
TX=0,RX=0,S=1	all cells marked as SHARED
TX=1,RX=0,S=1	all cells marked as RX and SHARED
TX=0,RX=1,S=1	all cells marked as TX and SHARED
TX=1,RX=1,S=1	all cells marked as TX and RX and SHARED

Figure 7: Meaning of the 6P CellOptions bitmap for the 6P COUNT and LIST requests.

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	Does not apply. 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 TXRX cells at B. TXRX cells at A.
TX=0,RX=0,S=1	Does not apply. 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 TXRX Shared cells at B. TXRX Shared cells at A.

Figure 8: Meaning of the 6P CellOptions bitmap for the 6P ADD, DELETE, RELOCATE 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 contents of the CellOptions field specify the options associated with all cells in the CellList. This necessarily means that the same options are associated with all cells in the CellList.

The 6P Cell is a 4-byte field, its RECOMMENDED format is:

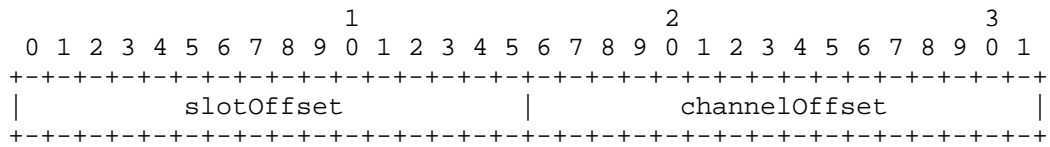


Figure 9: 6P Cell Format.

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 SF MAY redefine the format of the CellList field.

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.

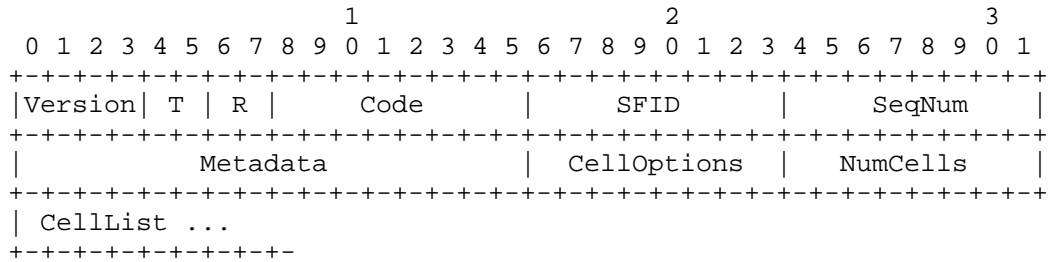


Figure 10: 6P ADD Request Format.

- 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 the sender wants to schedule to the receiver.
- CellList: A list of 0, 1 or multiple candidate cells.

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

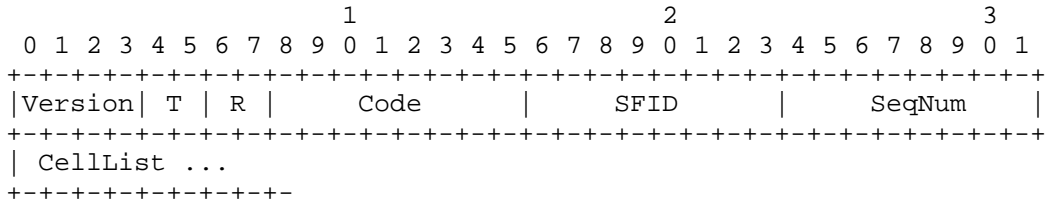


Figure 11: 6P ADD Response and Confirmation Formats.

CellList: A list of 0, 1 or multiple 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.

Upon receiving the request, 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 (less 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 0 elements (fail), NumCells elements (succeeded) or between 0 and NumCells elements (partially succeeded).

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

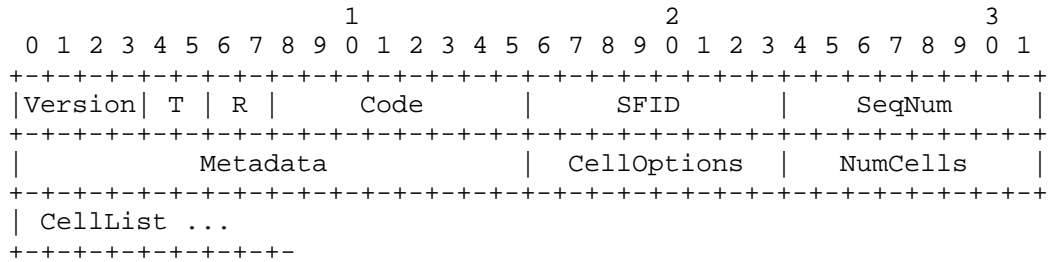


Figure 12: 6P DELETE Request Format.

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

CellOptions: Indicates the options that need to be associated to the cells to delete. Only the cells matching the CellOptions 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, 1 or multiple 6P Cells.

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

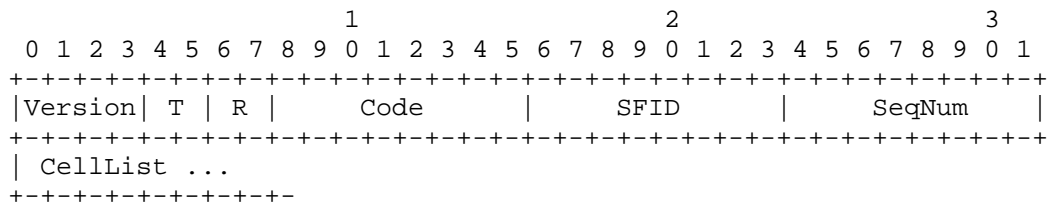


Figure 13: 6P DELETE Response and Confirmation Formats.

CellList: A list of 0, 1 or multiple 6P Cells.

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

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

- o All cells in the CellList MUST already be scheduled between the two nodes and MUST match the CellOptions field. If node A puts cells in its CellList that are not already scheduled between the two nodes and match the CellOptions field, node B MUST reply with a RC_ERR_CELLLIST return code.
- o If the CellList in the 6P Request is empty, the SF on the receiving node SHOULD delete any cell from the sender, as long as it matches the CellOptions field.
- o The CellList in a 6P Request (2-step transaction) or 6P Response (3-step transaction) MUST either be empty, contain exactly NumCells cells, or more than NumCells cells. The case where the CellList is not empty but contains less than NumCells cells is not supported.

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.

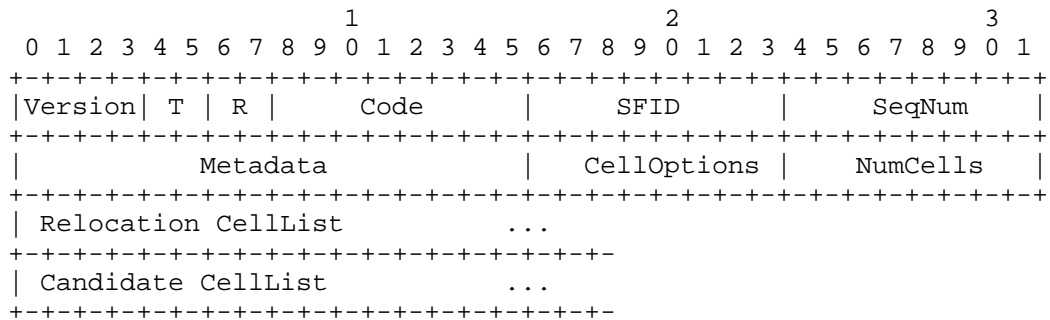


Figure 14: 6P RELOCATE Request Format.

- Metadata: Same usage as for the 6P ADD command, see Section 3.3.1.
- CellOptions: Indicates the options that need to be associated to the relocated cells.
- 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.

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.

In a 3-step 6P RELOCATE Transaction, node A only specifies 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.

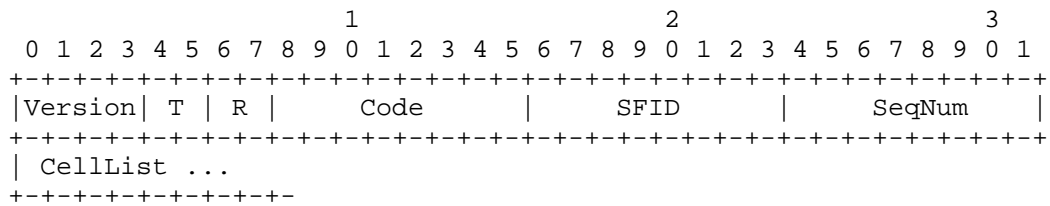


Figure 15: 6P RELOCATE Response and Confirmation Formats.

CellList: A list of 0, 1 or multiple 6P Cells.

Node A's SF wants to relocate NumCells cells. Node A creates a 6P RELOCATE Request, and indicates the cells to relocate in the Relocation CellList. It also selects NumCandidate cells from its schedule as candidate cells for node B, 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 indeed scheduled with node A, and are associate the options specified in the CellOptions field. If that check fails, node B MUST send a 6P Response to node A with return code RC_ERR_CELLLIST. If that check passes, 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 (less 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 were scheduled following the CellOptions field. That can contain 0 elements (when

the verification failed), NumCells elements (succeeded) or between 0 and NumCells elements (partially succeeded). If $N < \text{NumCells}$ cells appear in the CellList, this means 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 location specified in the CellList of the 6P Response. In case the received Response Code is RC_ERR_CELLLIST. The transaction is aborted and no cell is relocated.

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

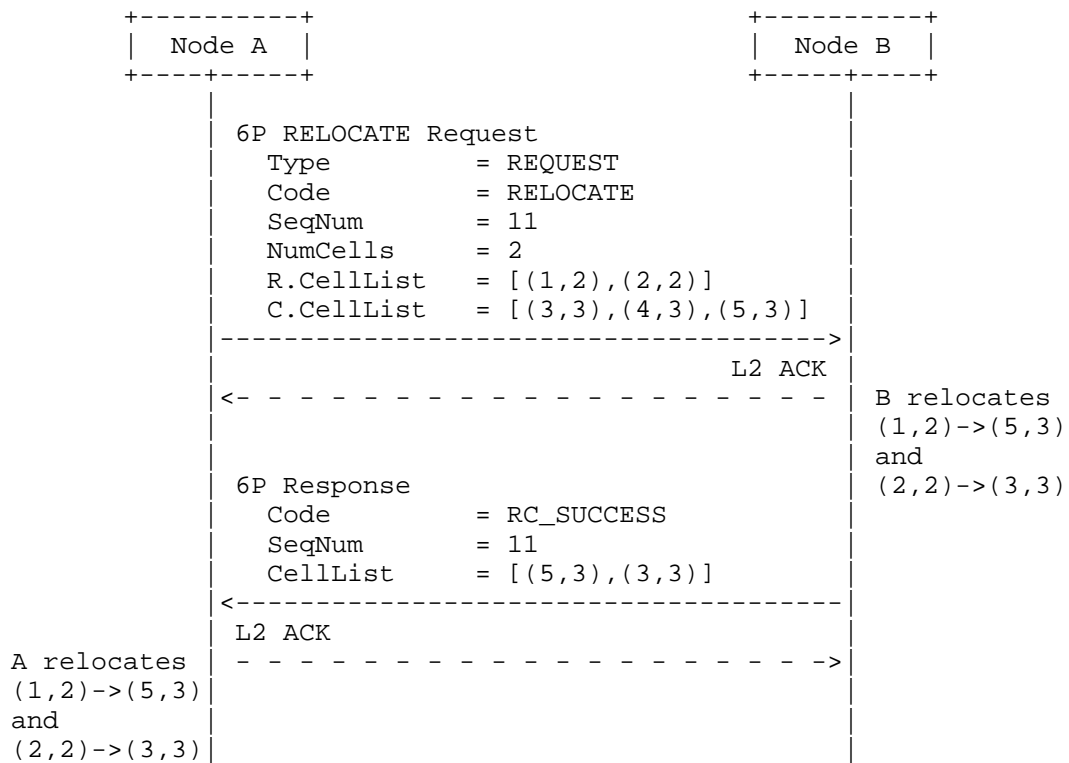


Figure 16: Example of a successful 2-step 6P RELOCATION Transaction.

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

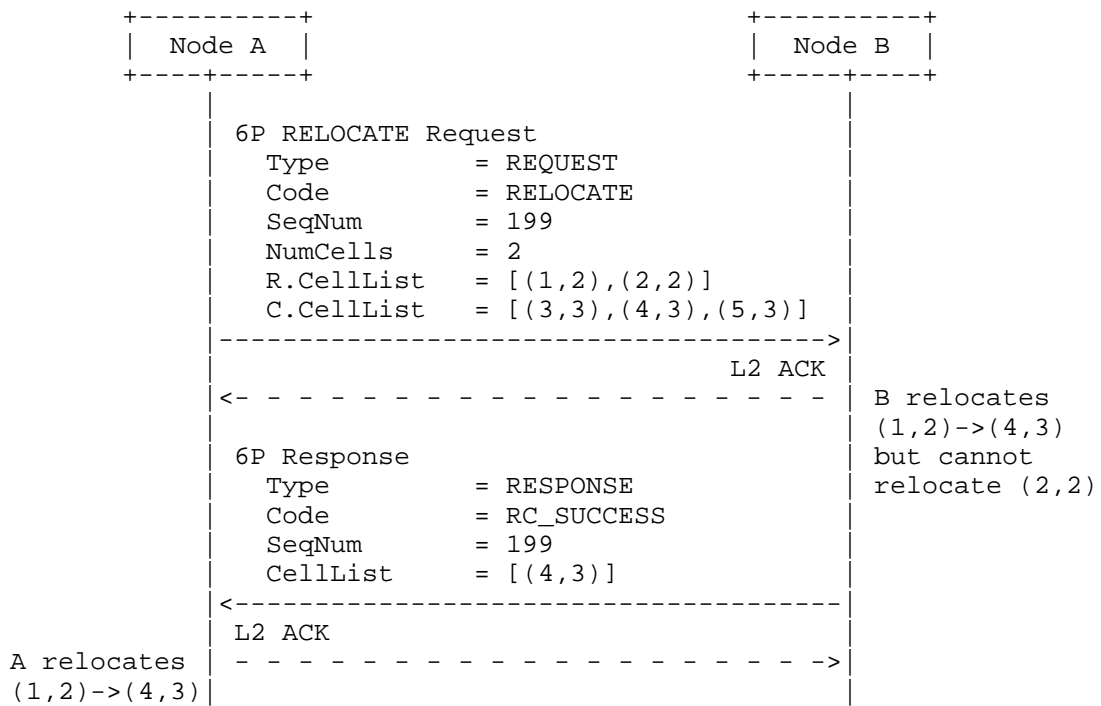


Figure 17: Example of a partially successful 2-step 6P RELOCATION Transaction.

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

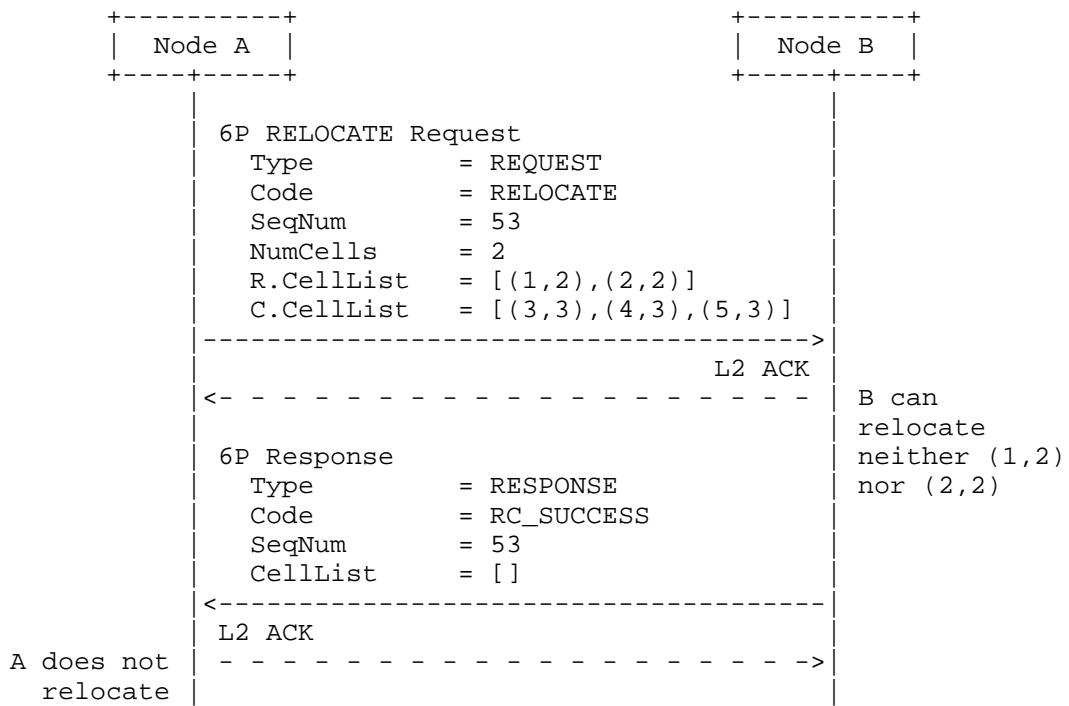


Figure 18: Failed 2-step 6P RELOCATION Transaction Example.

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

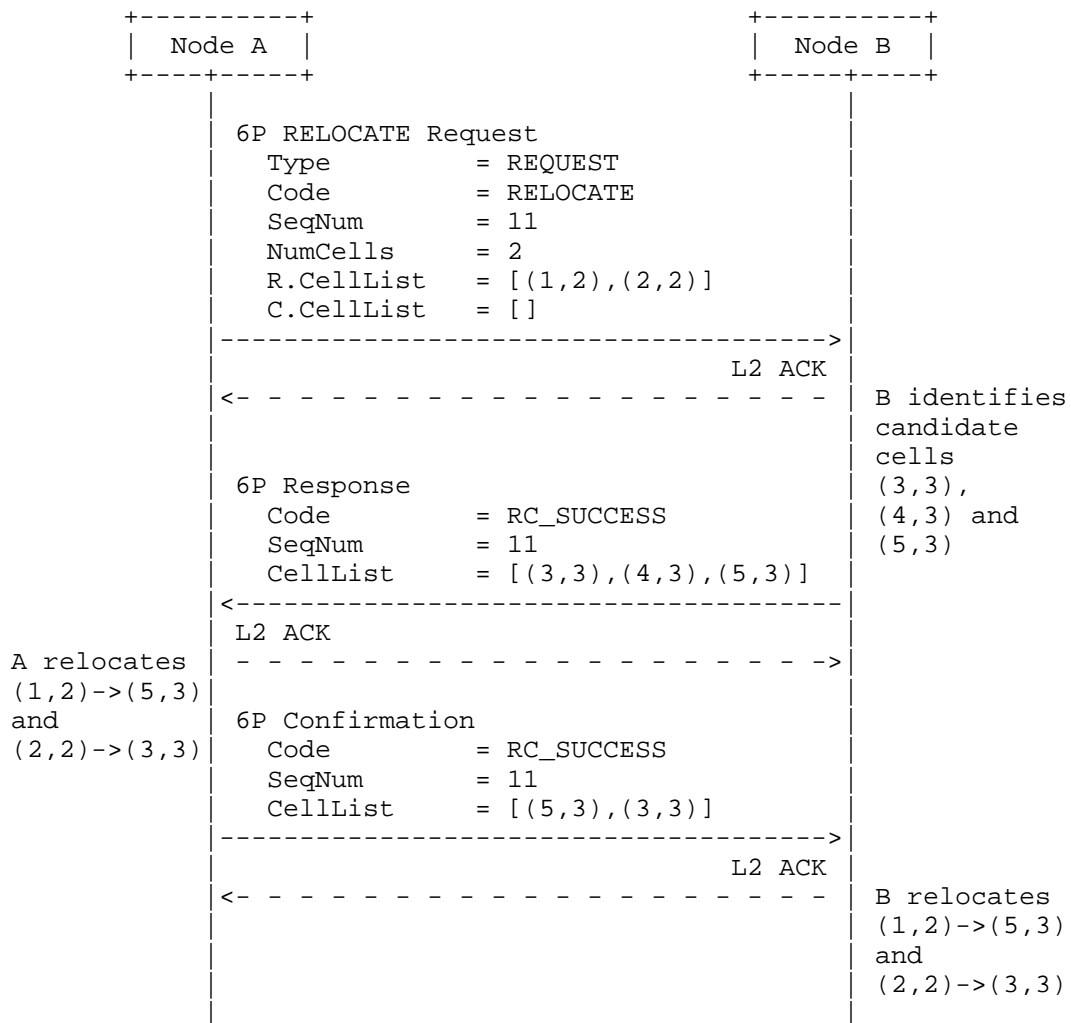


Figure 19: Example of a successful 3-step 6P RELOCATION Transaction.

3.3.4. Counting Cells

To retrieve the number of scheduled cells at 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.

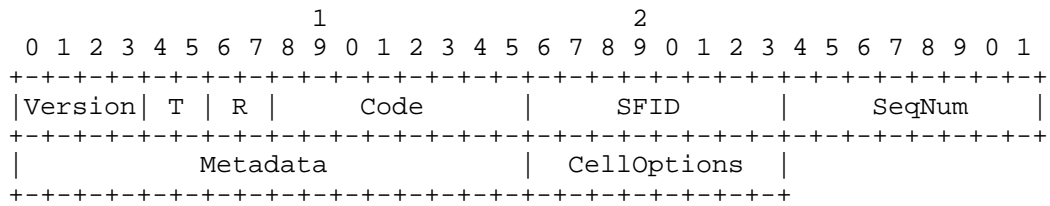


Figure 20: 6P COUNT Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1. Its format is the same as that in 6P ADD command, but its contents could be different.
 CellOptions: Specifies which types of cells to be counted.

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

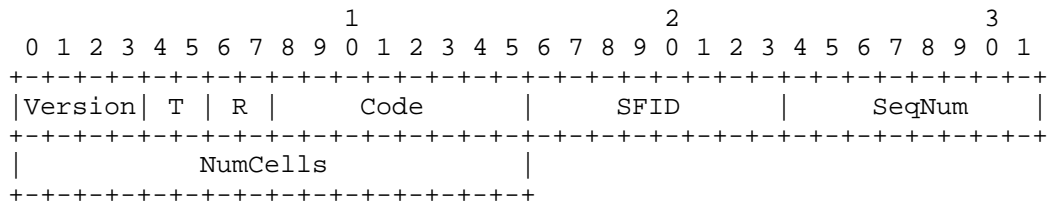


Figure 21: 6P COUNT Response Format.

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

Node A issues a COUNT command to node B, specifying a set of 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, and 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 at 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.

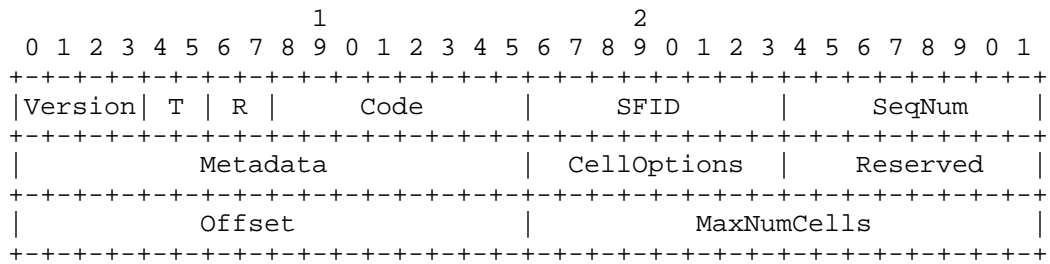


Figure 22: 6P LIST Request Format.

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

CellOptions: Specifies which types of cells 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 less than MaxNumCells cells, for example if MaxNumCells cells do not fit in the frame.

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

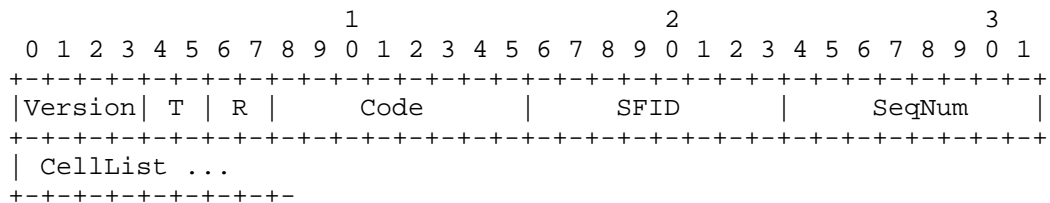


Figure 23: 6P LIST Response Format.

CellList: A list of 0, 1 or multiple 6P Cells.

When receiving a LIST command, node B returns the cells 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 1 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 (as per Figure 37), 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 less 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.

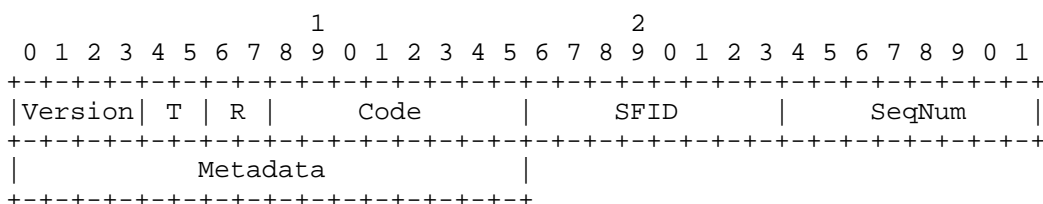


Figure 24: 6P CLEAR Request Format.

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

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

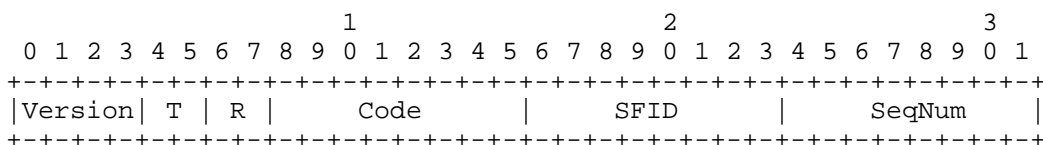


Figure 25: 6P CLEAR Response Format.

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 mismatch, the transaction MUST NOT be aborted. Upon 6P CLEAR completion, the value of SeqNum MUST be reset to 0.

The Response Code to a 6P CLEAR command SHOULD be RC_SUCCESS unless the operation cannot be executed. When the CLEAR operation cannot be executed the Response 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. 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.

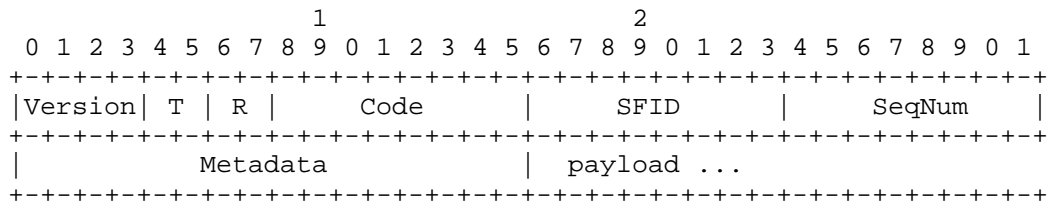


Figure 26: 6P SIGNAL Request Format.

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

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

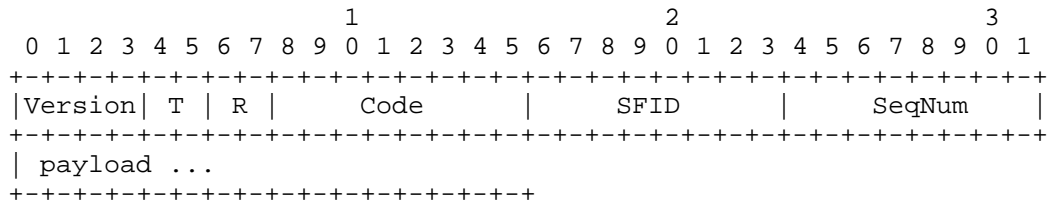


Figure 27: 6P SIGNAL Response Format.

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.

In case 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 in 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 and a 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 6P Response in 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 having received the 6P Response for a previous request to that neighbor, except when the previous 6P Transaction has 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 37) and discard this ongoing second transaction. A node receiving RC_RESET code MUST abort the second transaction and consider it never happened.

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 from 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 37). 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 37). 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 sending the 6P Request has not received the 6P Response within a specified amount of time determined by the SF. In a 3-step transaction, a timeout also occurs when the node sending the 6P Response has not received the 6P Confirmation. When a timeout occurs the transaction MUST be cancelled at the node where the timeout occurred. The value of the 6P Timeout should be larger than the longest possible time it can take for the exchange to finish. 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 it 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 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, that 6P Request will have a different SeqNum.

Similarly, 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) .

The SeqNum MUST be implemented as a lollipop counter: it rolls over from 0xFF to 0x01 (not to 0x00). This is used to detect that 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 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.

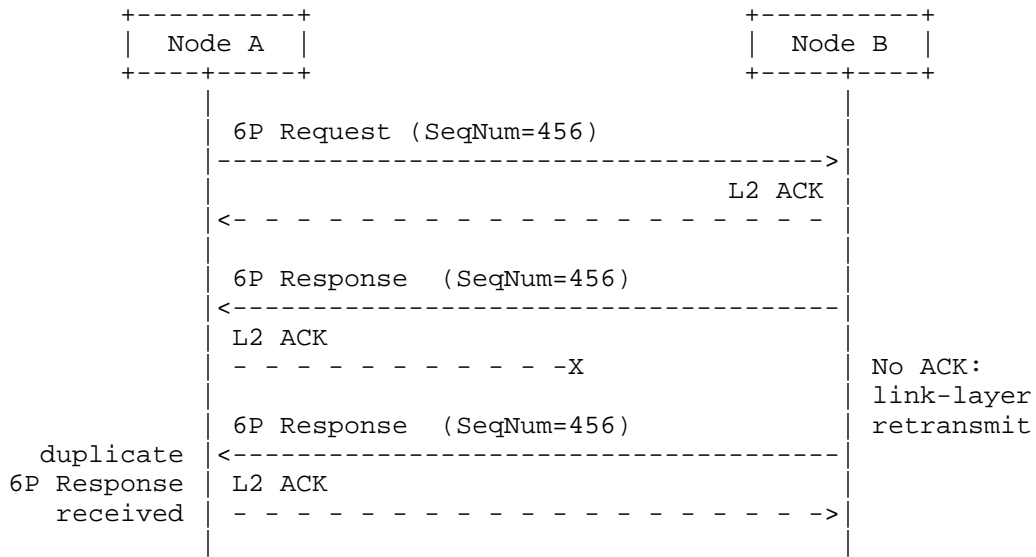


Figure 29: Example duplicate 6P message.

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.

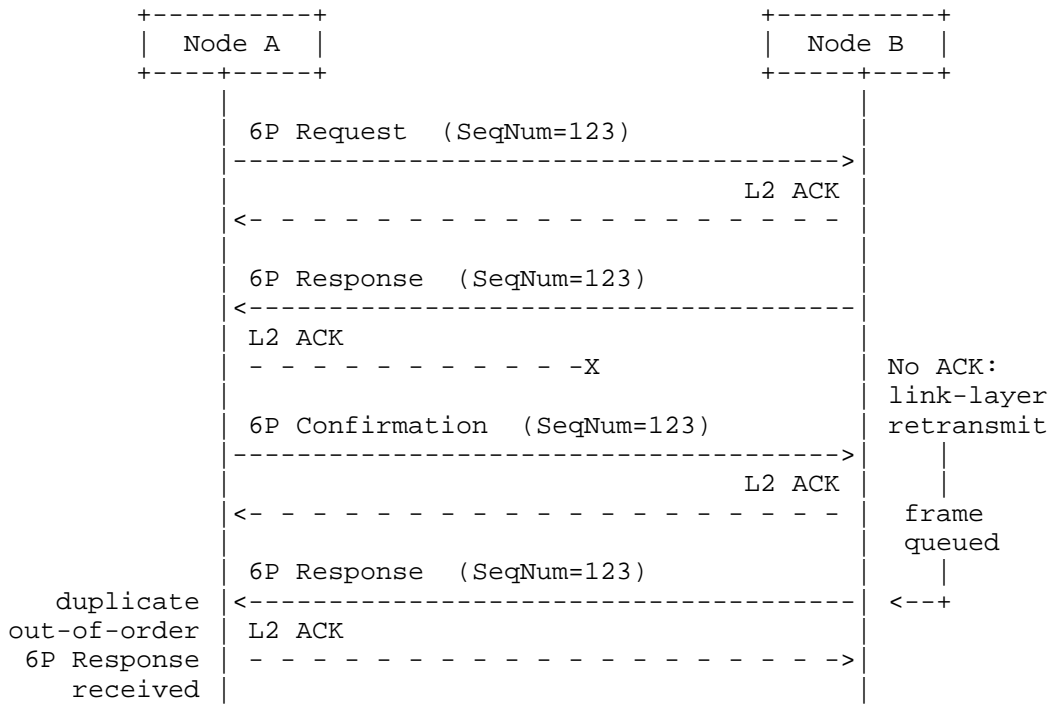


Figure 30: Example out-of-order duplicate 6P message.

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 it 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 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. Given that the SeqNum field increments by 1 at each message. 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 on, 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 power cycled. 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.

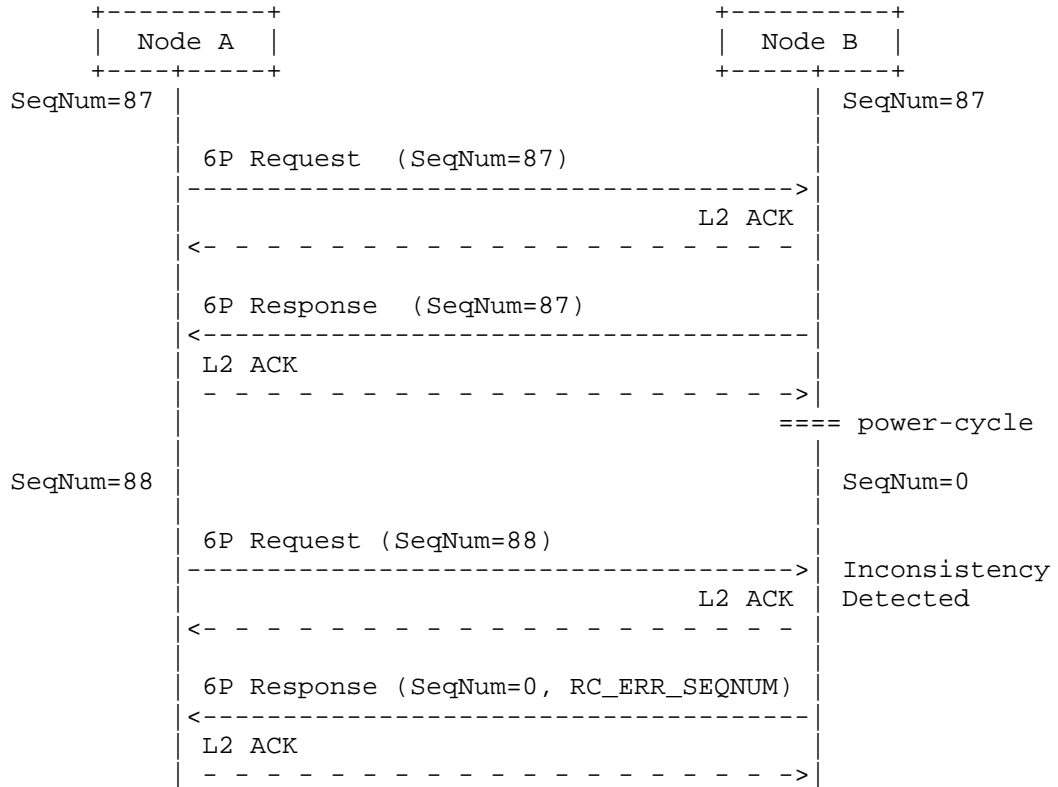


Figure 31: Example of inconsistency because of node reset.

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 32.

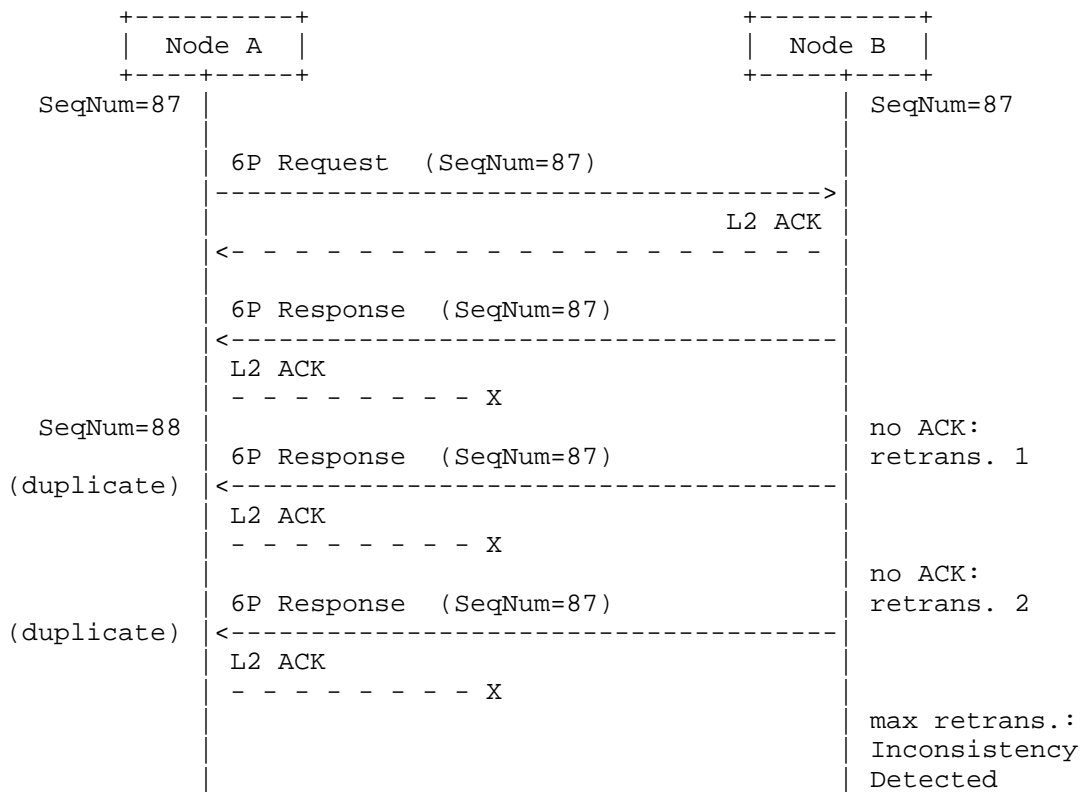


Figure 32: Example inconsistency because of maximum link-layer retransmissions (here 2).

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 (to 0 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 37 indicates an error. When a node receives a 6P Response or 6P Confirmation with such an error, it MUST consider the 6P Transaction as failed. In particular, if this was a response to a 6P ADD/DELETE/RELOCATE Request, the node MUST NOT add/delete/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. 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 are secured through link-layer security. When link-layer security is enabled, the 6P messages MUST be secured. This is possible because 6P messages are carried as Payload IE.

4. Requirements for 6top Scheduling Functions (SF)

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

The specification for an SF

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

- o MUST specify the list of statistics to gather. An example statistic is the number of transmitted frames to each neighbor. In case the SF requires no statistics to be gathered, the specific of the SF MUST explicitly state so.
- o SHOULD clearly state the application domain the SF is created for.
- o SHOULD contain examples which highlight normal and error scenarios.
- o SHOULD contain a list of current implementations, at least during the I-D state of the document, per [RFC6982].
- o SHOULD contain a performance evaluation of the scheme, possibly through references to external documents.
- o SHOULD define the format of the SIGNAL command payload and its use.
- o MAY redefine the format of the CellList field.
- o MAY redefine the format of the CellOptions field.
- o MAY redefine the meaning of the CellOptions field.

5. Security Considerations

6P messages are carried inside 802.15.4 Payload Information Elements (IEs). Those Payload IEs are encrypted and authenticated at the link layer through CCM* [CCM-Star] 6P benefits from the same level of security as any other Payload IE. The 6P protocol does not define its own security mechanisms. The 6P protocol does not provide protection against DOS attacks. This is relevant in 3-step transactions when a confirmation message could not be sent in purpose by the attacker. Such situations SHOULD be handled by an appropriate policy such as blacklisting the attacker after several attempts. Other DoS attacks are possible by sending unmeaningful requests to nodes. The effect to the overall network can be minimal as communication between attacked node and attacker happen in dedicated cells. DoS then only limits that cells. Yet, this can be avoided by blacklisting the node after several attempts. When to blacklist is policy specific and SHOULD be addressed by the SF. A key management solution is out of scope for this document. The 6P protocol will benefit for the key management solution used in the network.

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]:

Subtype	Name	Reference
IANA_6TOP_SUBIE_ID	6P	RFCXXXX

Figure 33: IETF IE Subtype '6P'.

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 34: 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 the 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 35: 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 the 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 36: 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 the 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 37: 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 the SFID in the range 0-255, the corresponding Name, and a reference to the 6P Scheduling Function's documentation.

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

SFID	Name	Reference
0	Minimal Scheduling Function (MSF)	draft-chang-6tisch-msf
1	Experimental Scheduling Function (SFX)	draft-ietf-6tisch-6top-sfx

Figure 38: 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 39. These specifications must follow the guidelines of Section 4.

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

Figure 39: SF Identifier (SFID): Registration Procedures.

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 the 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 40: 6P CellOptions bitmap.

All other Message Types are Reserved.

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

7. References

7.1. Normative References

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

7.2. Informative References

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

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

Appendix A. Recommended Structure of an SF Specification

The following section structure for a SF document is RECOMMENDED:

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

Appendix B. Implementation Status

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

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

First F-Interop ETSI 6TiSCH plugtests: 6P is one of the protocols addressed during the First F-Interop ETSI 6TiSCH plugtests

organized on 14-15 July 2017 in Prague, Czech Republic. It was attended by 14 entities, which 4-5 independent implementation bases.

ETSI 6TiSCH/6lo plugtests: 6P was one of the protocols addressed during the ETSI 6TiSCH #3 plugtests organized on 15-17 July 2016 in Berlin, Germany. 15 entities participated in this event, verifying the compliance and interoperability of their implementation of 6P. This event happened under NDA, so neither the name of the entities nor the test results are public. This event is, however, a clear indication of the maturity of 6P, and the interest it generates. More information about the event at <http://www.etsi.org/news-events/events/1077-6tisch-6lo-plugtests>.

ETSI 6TiSCH #2 plugtests: 6P was one of two protocols addressed during the ETSI 6TiSCH #2 plugtests organized on 2-4 February 2016 in Paris, France. 14 entities participated in this event, verifying the compliance and interoperability of their implementation of 6P. This event happened under NDA, so neither the name of the entities nor the test results are public. This event is, however, a clear indication of the maturity of 6P, and the interest it generates. More information about the event at <http://www.etsi.org/news-events/events/1022-6TiSCH-2-plugtests>.

OpenWSN: 6P is implemented in the OpenWSN project [OpenWSN] under a BSD open-source license. The authors of this document are collaborating with the OpenWSN community to gather feedback about the status and performance of the protocols described in this document. Results from that discussion will appear in this section in future revision of this specification. More information about this implementation at <http://www.openwsn.org/>.

F-Interop Interoperability/Conformance Testing tool The F-Interop project is putting together an online tool to conduct online and remote interoperability/conformance tests. 6P is one of the supported protocols.

6TiSCH simulator The 6TiSCH simulator is a Python-based high-level simulator which implements 6P and is built to evaluate the performance of different SFs. More information at <https://bitbucket.org/6tisch/simulator/>.

Wireshark Dissector: A Wireshark dissector for 6P is implemented under a BSD open-source license. It is developed and maintained at <https://github.com/openwsn-berkeley/dissectors/>, and regularly merged into the main Wireshark repository. Please see the Wireshark documentation to see what version of 6P it supports.

Appendix C. [TEMPORARY] Changelog

- o draft-ietf-6tisch-6top-protocol-10

- * Adding a table detailing celloptions usage in ADD/DELETE/RELOCATE operations.

- * Addressing comments from IoT Directorate.
- * Fixing typos.
- o draft-ietf-6tisch-6top-protocol-09
 - * Requiring version 0 in RC_ERR_VERSION response.
 - * Adding L2 ACK in figures.
 - * Inconsistency management update.
 - * Moving SF requirements to another section.
 - * Moving implementation status to appendix.
 - * Fixing typos.
- o draft-ietf-6tisch-6top-protocol-08
 - * Replacing GEN counter by SeqNum and timeout.
 - * Adding SIGNAL command.
 - * Adding RC_ERR_SEQNUM return code.
 - * Clarifying IETF IE usage.
 - * Cleaning up error codes.
 - * Fixing typos.
- o draft-ietf-6tisch-6top-protocol-07
 - * Inverting RC_ERR_LOCKED and RC_ERR_BUSY error codes for concurrent transactions.
 - * Adding missing implementations.
 - * Fixing references.
 - * Fixing typos.
- o draft-ietf-6tisch-6top-protocol-06
 - * Changing error code from RC_RESET to RC_ERR_CELLLIST when deleting unscheduled cells.
 - * Fixing typos.
- o draft-ietf-6tisch-6top-protocol-05
 - * complete reorder of sections. Merged protocol behavior and command description
 - * STATUS to COUNT
 - * written-out IANA section
 - * complete proof-read
- o draft-ietf-6tisch-6top-protocol-04
 - * recommendation on which cells to use for 6P traffic
 - * relocation format: added numberOfCells field
 - * created separate section about "cell suggestion"
 - * Added RC_ERR_CELLLIST and RC_ERR_EOL error codes
 - * Added example for two step with the failure
 - * Recommended numbers in IANA section
 - * single generation number
 - * IEEE802.15.4 -> IEEE Std 802.15.4 or 802.15.4
 - * complete proof-read

- o draft-ietf-6tisch-6top-protocol-03
 - * Added a reference to [RFC8137].
 - * Added the Type field.
 - * Editorial changes (figs, typos, ...)
- o draft-ietf-6tisch-6top-protocol-02
 - * Rename COUNT to STATUS
 - * Split LIST to LIST AB and LIST BA
 - * Added generation counters and describing generation tracking of the schedule
 - * Editorial changes (figs, typos, ...)
- o draft-ietf-6tisch-6top-protocol-01
 - * Clarifying locking of resources in concurrent transactions
 - * Clarifying return of RC_ERR_BUSY in case of concurrent transactions without enough resources
- o draft-ietf-6tisch-6top-protocol-00
 - * Informational to Std track
- o draft-wang-6tisch-6top-protocol-00
 - * Editorial overhaul: fixing typos, increasing readability, clarifying figures.
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/47>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/54>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/55>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/49>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/53>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/44>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/48>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/43>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/52>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/45>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/51>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/50>

- * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/46>
- * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/41>
- * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/42>
- * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/39>
- * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/40>
- o draft-wang-6tisch-6top-sublayer-05
 - * Specifies format of IE
 - * Adds token in messages to match request and response
- o draft-wang-6tisch-6top-sublayer-04
 - * Renames IANA_6TOP_IE_GROUP_ID to IANA_IETF_IE_GROUP_ID.
 - * Renames IANA_CMD and IANA_RC to IANA_6TOP_CMD and IANA_6TOP_RC.
 - * Proposes IANA_6TOP_SUBIE_ID with value 0x00 for the 6top sub-IE.
- o draft-wang-6tisch-6top-sublayer-03
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/32/missing-command-list>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/31/missing-command-count>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/30/missing-command-clear>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/37/6top-atomic-transaction-6p-transaction>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/35/separate-opcode-from-rc>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/36/add-length-field-in-ie>
 - * https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/27/differentiate-rc_err_busy-and
 - * https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/29/missing-rc-rc_reset
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/28/the-sf-must-specify-the-behavior-of-a-mote>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/26/remove-including-their-number>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/34/6of-sf>
 - * <https://bitbucket.org/6tisch/draft-wang-6tisch-6top-protocol/issues/33/add-a-figure-showing-the-negotiation>
- o draft-wang-6tisch-6top-sublayer-02

- * introduces the 6P protocol and the notion of 6top Transaction.
- * introduces the concept of 6OF and its 6OFID.

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: Standards Track
Expires: December 22, 2018

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

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

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 22, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

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

1. Introduction

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

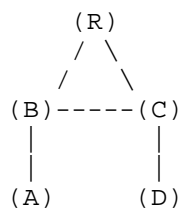


Figure 1: A simple 6TiSCH network.

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

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

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

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

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

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

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

2. 6TiSCH Operation Sublayer (6top)

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

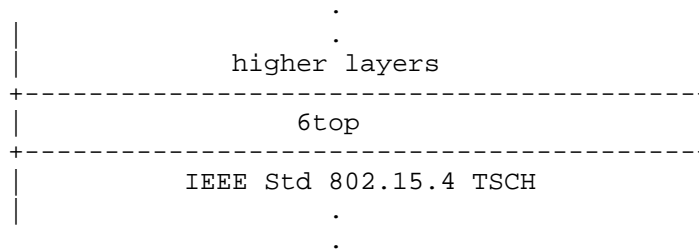


Figure 2: The 6top sublayer in the protocol stack.

The roles of the 6top sublayer are to:

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

2.1. Hard/Soft Cells

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

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

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

2.2. Using 6P with the Minimal 6TiSCH Configuration

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

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

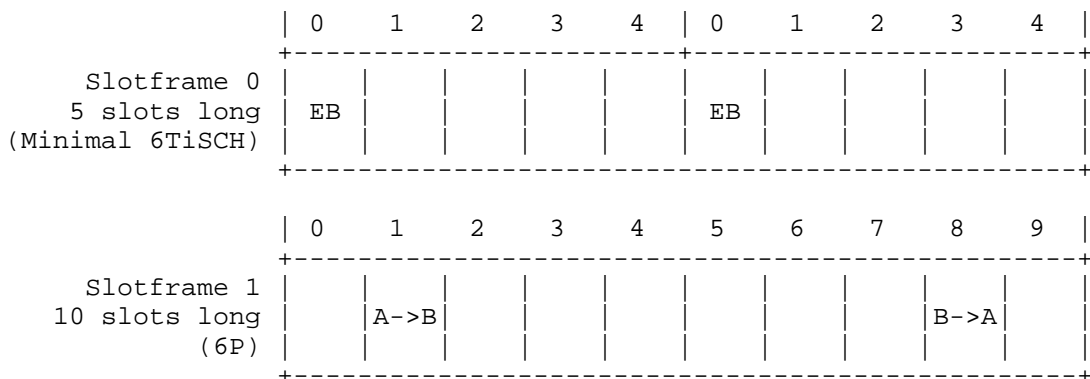


Figure 3: 2-slotframe structure when using 6P alongside the Minimal 6TiSCH Configuration.

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

3. 6top Protocol (6P)

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

3.1. 6P Transactions

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

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

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

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

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

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

3.1.1.1. 2-step 6P Transaction

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

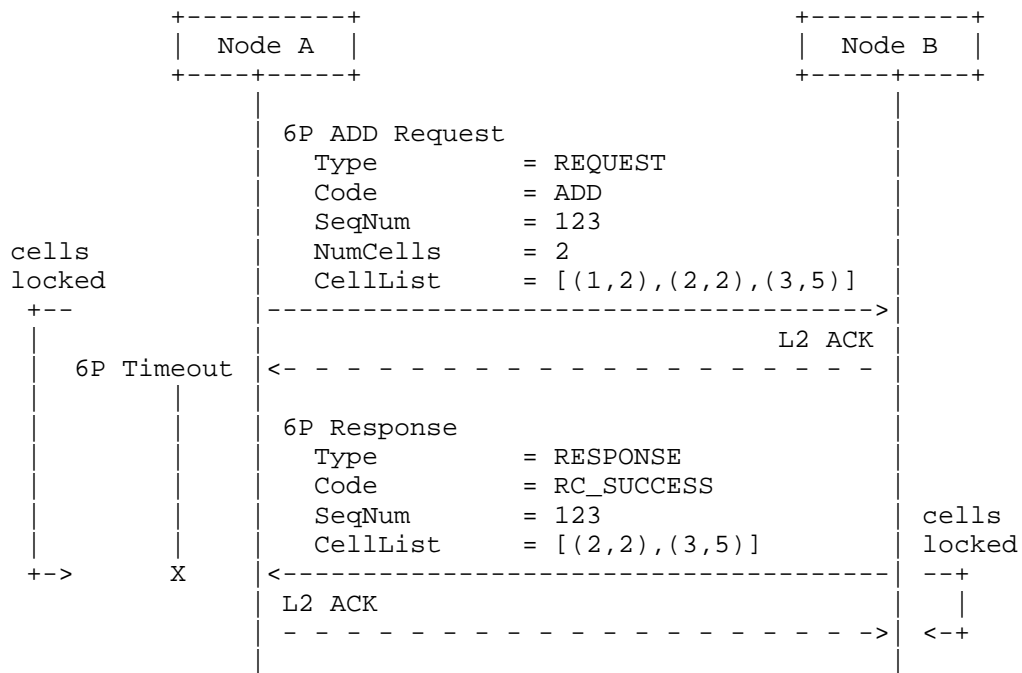


Figure 4: An example 2-step 6P Transaction.

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

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

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

3.1.2. 3-step 6P Transaction

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

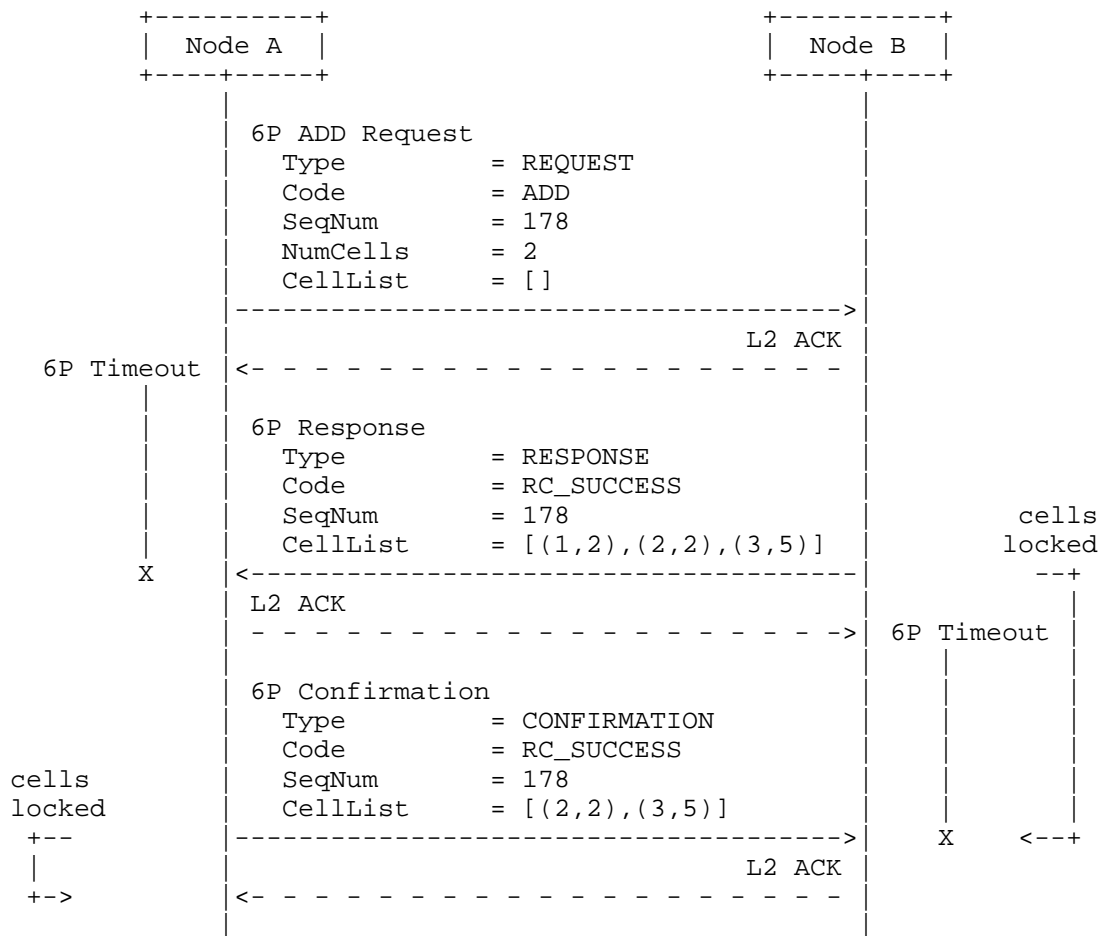


Figure 5: An example 3-step 6P Transaction.

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

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

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

3.2. Message Format

3.2.1. 6top Information Element (IE)

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

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

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

3.2.2. Generic 6P Message Format

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

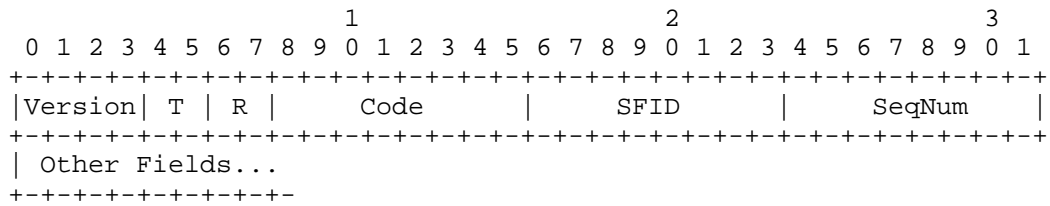


Figure 6: Generic 6P Message Format.

- 6P Version (Version): The version of the 6P protocol. Only version 0 is defined in this document. Future specifications may define further versions of the 6P protocol.
- Type (T): Type of message. The message types are defined in Section 6.2.2.
- Reserved (R): Reserved bits. These two bits SHOULD be set to zero when sending the message, and MUST be ignored upon reception.
- Code: The Code field contains a 6P Command Identifier when the 6P message is of Type REQUEST. Section 6.2.3 lists the 6P command identifiers. The Code field contains a 6P return code when the 6P message is of Type RESPONSE or CONFIRMATION. Section 6.2.4 lists the 6P return codes. The same return codes are used in both 6P Response and 6P Confirmation messages.
- 6top Scheduling Function Identifier (SFID): The identifier of the SF to use to handle this message. The SFID is defined in Section 4.1.
- SeqNum: Sequence number associated with the 6P Transaction, used to match the 6P Request, 6P Response and 6P Confirmation of the same 6P Transaction. The value of SeqNum MUST be different at each new 6P Request issued to the same neighbor and using the same SF. The SeqNum is also used to ensure consistency between the schedules of the two neighbors. Section 3.4.6 details how the SeqNum is managed.
- Other Fields: The list of other fields and how they are used is detailed in Section 3.3.

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

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

3.2.3. 6P CellOptions

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

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

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

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

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

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

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

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

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

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

3.2.4. 6P CellList

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

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

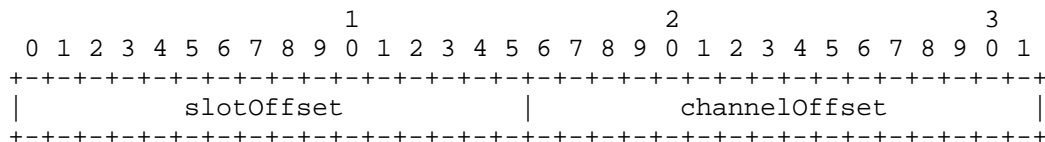


Figure 9: 6P Cell Format.

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

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

3.3. 6P Commands and Operations

3.3.1. Adding Cells

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

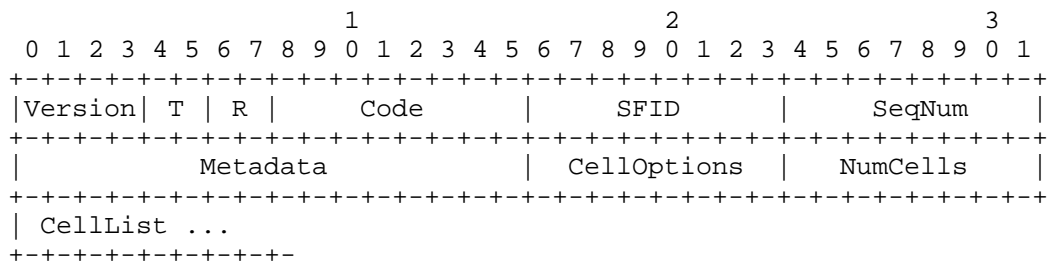


Figure 10: 6P ADD Request Format.

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

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

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

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

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

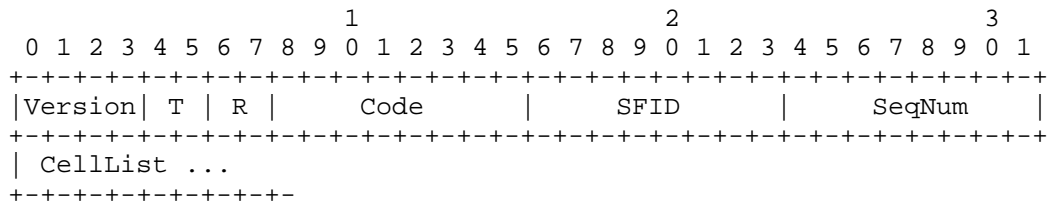


Figure 11: 6P ADD Response and Confirmation Formats.

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

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

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

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

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

3.3.2. Deleting Cells

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

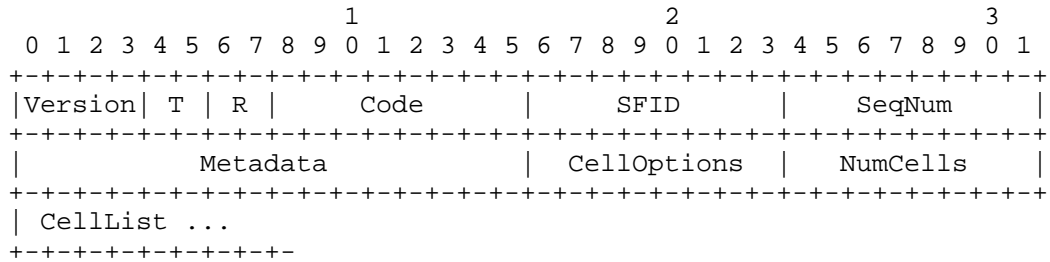


Figure 12: 6P DELETE Request Format.

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

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

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

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

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

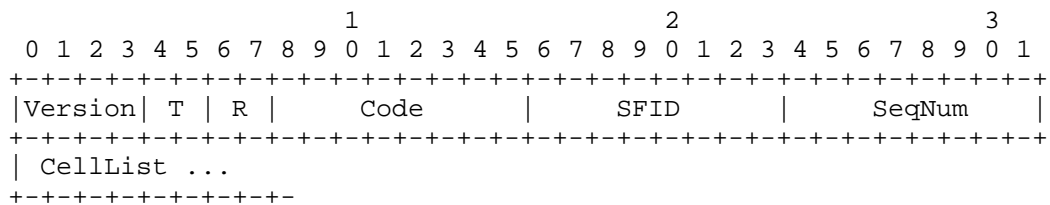


Figure 13: 6P DELETE Response and Confirmation Formats.

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

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

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

3.3.3. Relocating Cells

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

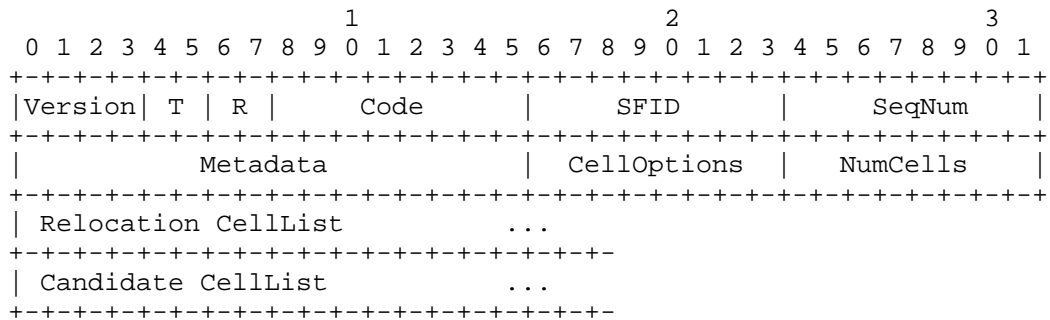


Figure 14: 6P RELOCATE Request Format.

- Metadata: Same usage as for the 6P ADD command, see Section 3.3.1.
- CellOptions: Indicates the options that need to be associated with cells to be relocated.
- NumCells: The number of cells to relocate, which MUST be equal or greater than 1.
- Relocation CellList: The list of NumCells 6P Cells to relocate.
- Candidate CellList: A list of NumCandidate candidate cells for node B to pick from. NumCandidate MUST be 0, equal to NumCells, or

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

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

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

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

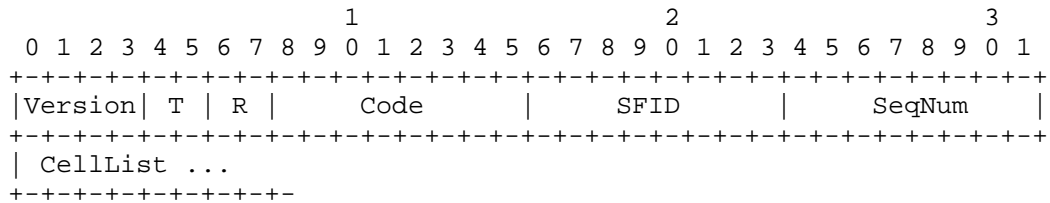


Figure 15: 6P RELOCATE Response and Confirmation Formats.

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

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

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

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

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

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

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

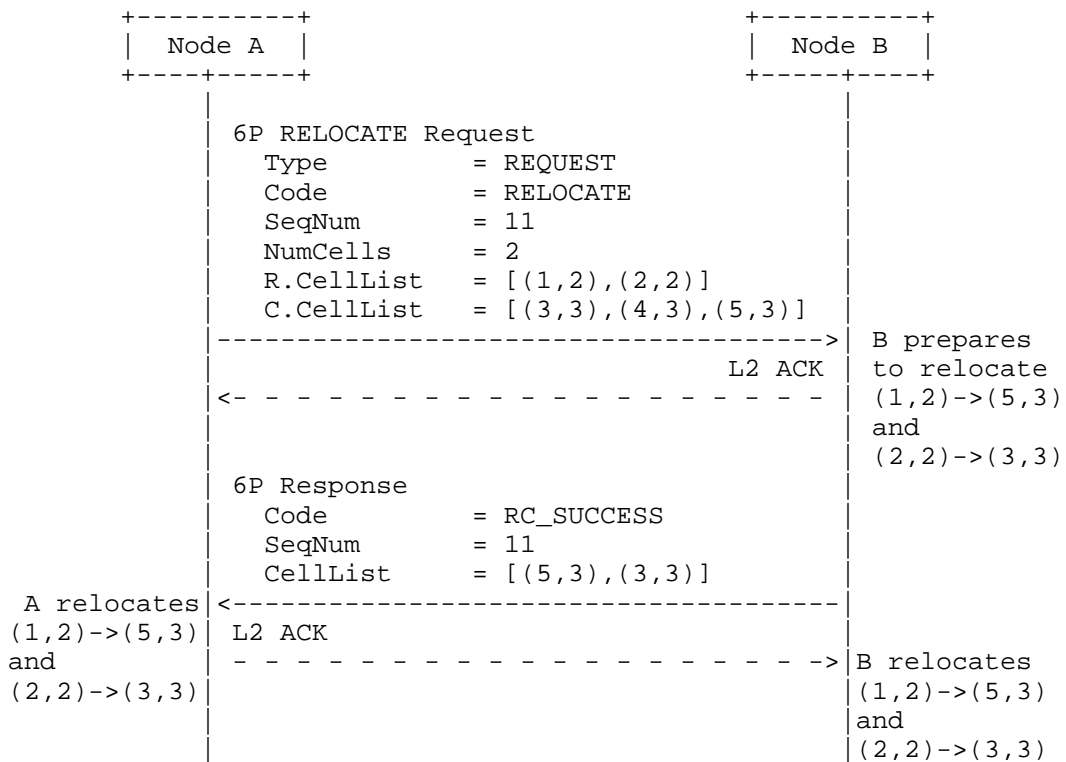


Figure 16: Example of a successful 2-step 6P RELOCATION Transaction.

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

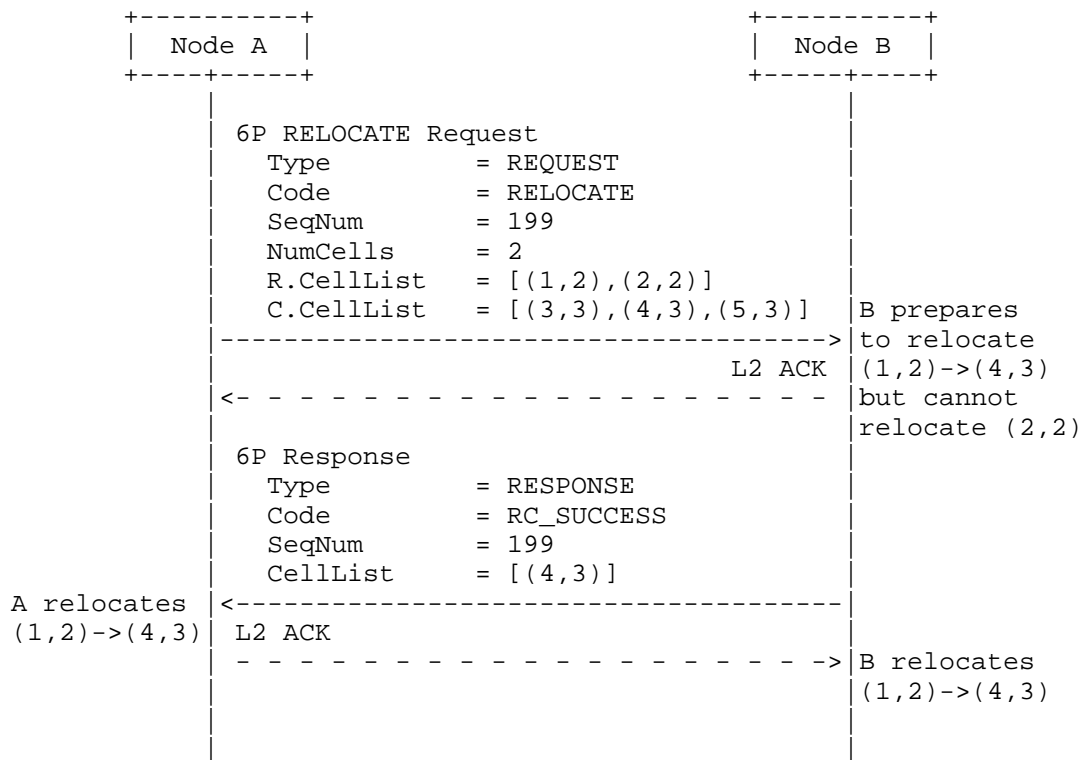


Figure 17: Example of a partially successful 2-step 6P RELOCATION Transaction.

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

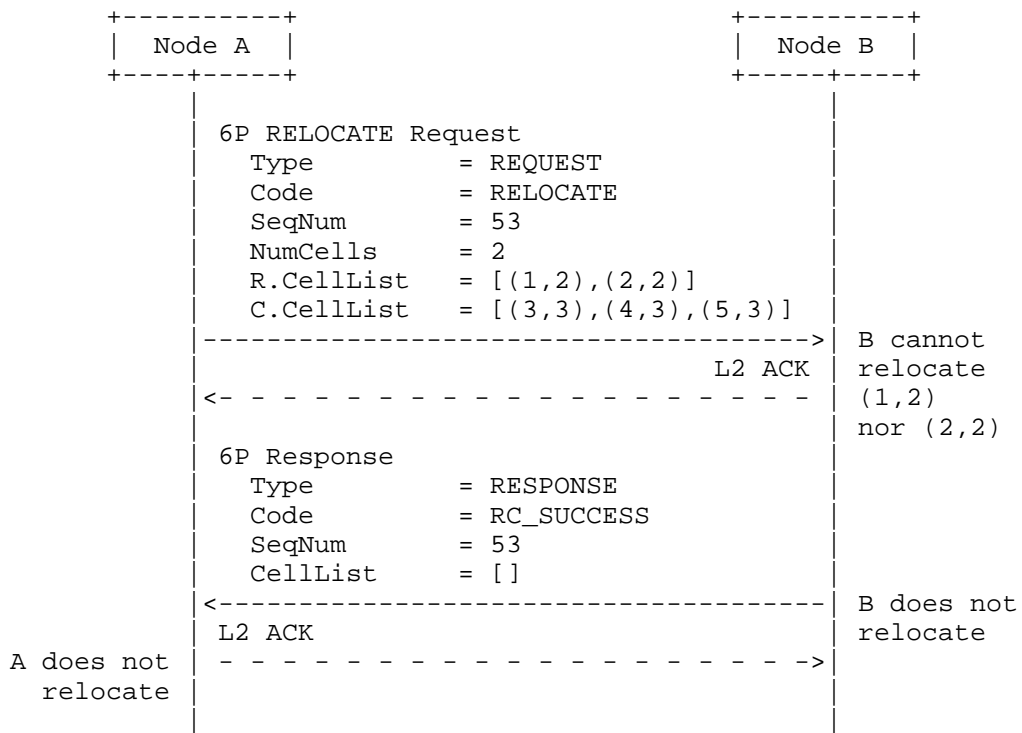


Figure 18: Failed 2-step 6P RELOCATION Transaction Example.

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

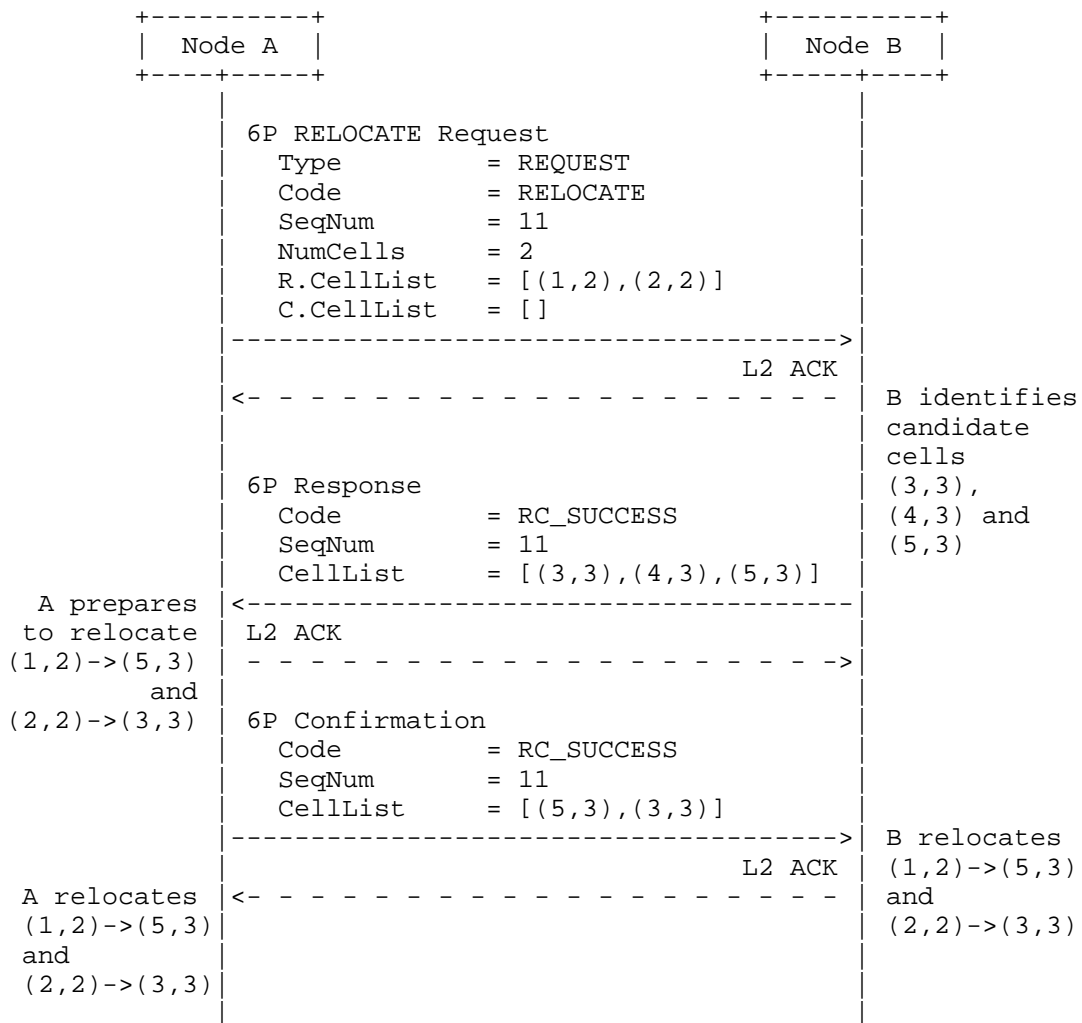


Figure 19: Example of a successful 3-step 6P RELOCATION Transaction.

3.3.4. Counting Cells

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

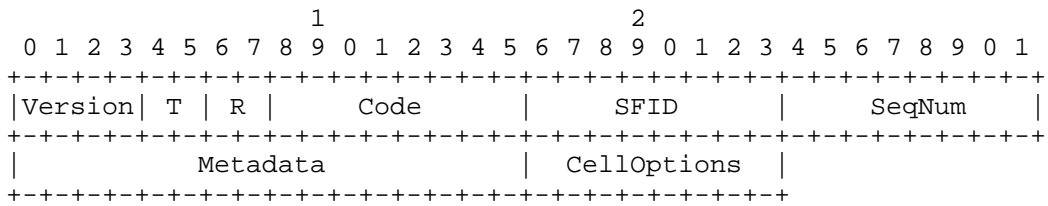


Figure 20: 6P COUNT Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 3.3.1. Its format is the same as that in the 6P ADD command, but its content could be different.
 CellOptions: Specifies which type of cell to be counted.

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

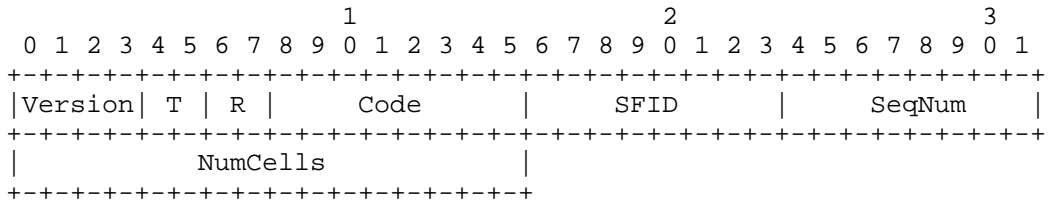


Figure 21: 6P COUNT Response Format.

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

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

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

3.3.5. Listing Cells

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

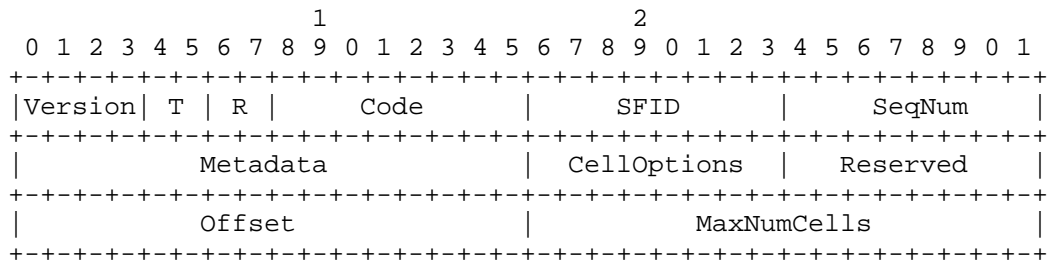


Figure 22: 6P LIST Request Format.

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

CellOptions: Specifies which type of cell to be listed.

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

Offset: The Offset of the first scheduled cell that is requested. The mechanism assumes cells are ordered according to a rule defined in the SF. The rule MUST always order the cells in the same way.

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

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

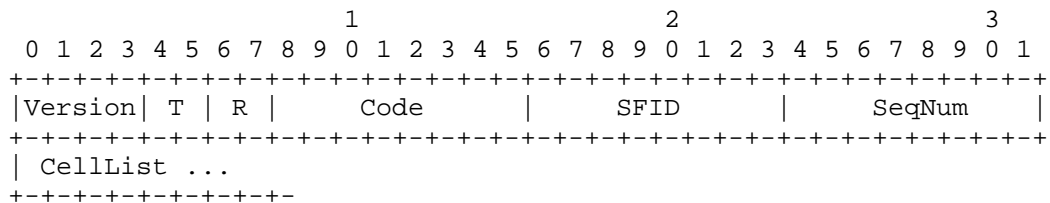


Figure 23: 6P LIST Response Format.

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

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

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

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

3.3.6. Clearing the Schedule

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

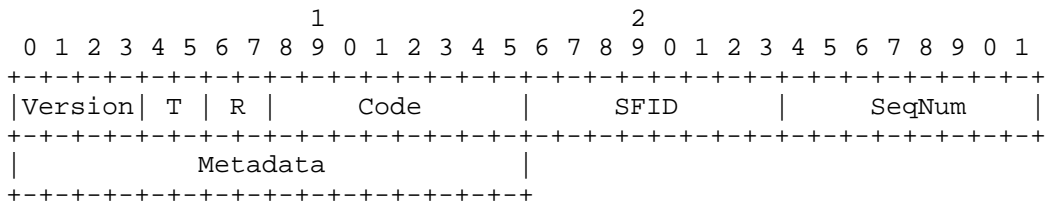


Figure 24: 6P CLEAR Request Format.

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

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

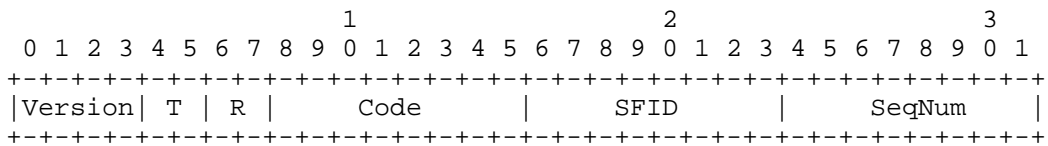


Figure 25: 6P CLEAR Response Format.

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

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

3.3.7. Generic Signaling Between SFs

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

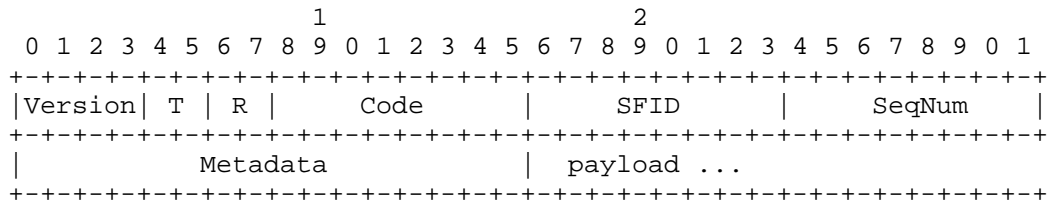


Figure 26: 6P SIGNAL Request Format.

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

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

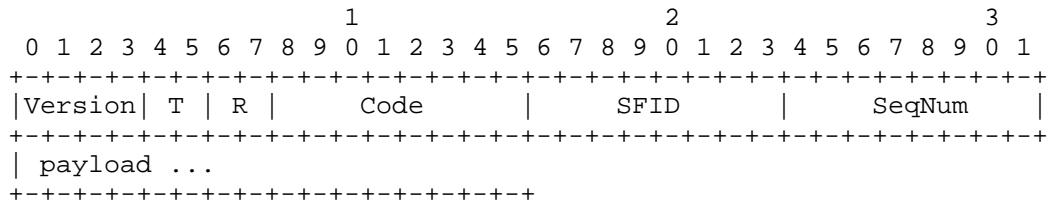


Figure 27: 6P SIGNAL Response Format.

3.4. Protocol Functional Details

3.4.1. Version Checking

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

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

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

3.4.2. SFID Checking

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

3.4.3. Concurrent 6P Transactions

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

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

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

3.4.4. 6P Timeout

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

3.4.5. Aborting a 6P Transaction

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

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

3.4.6. SeqNum Management

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

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

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

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

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

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

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

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

Figure 28: Possible values of the SeqNum.

3.4.6.1. Detecting and Handling Duplicate 6P Messages

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

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

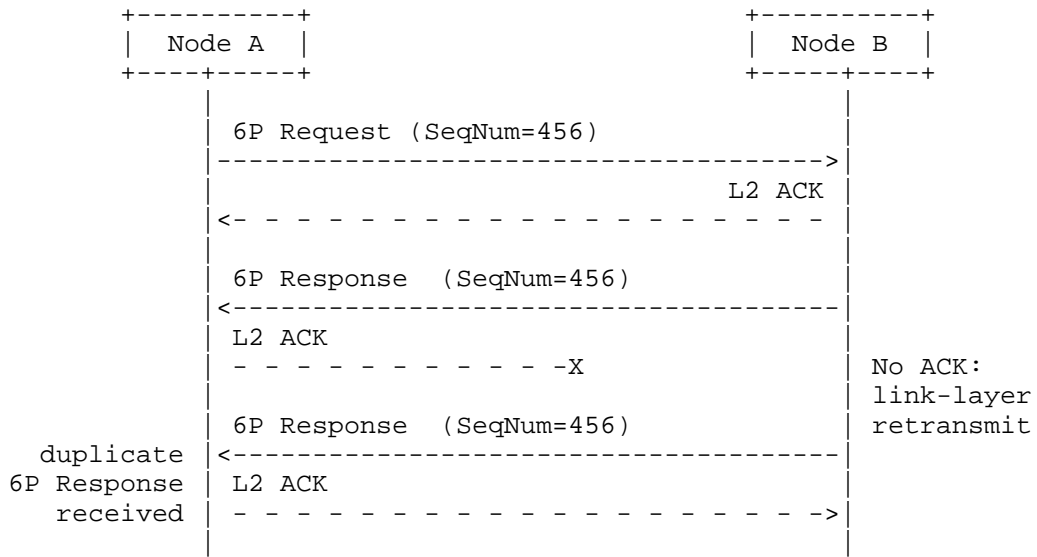


Figure 29: Example duplicate 6P message.

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

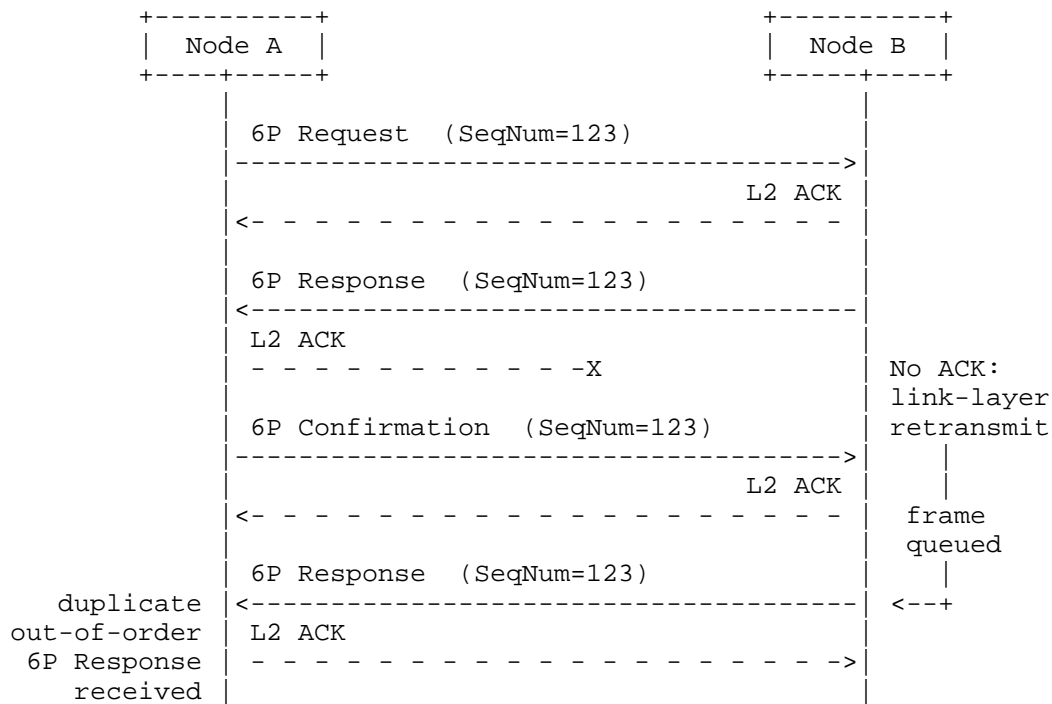


Figure 30: Example out-of-order duplicate 6P message.

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

3.4.6.2. Detecting and Handling a Schedule Inconsistency

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

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

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

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

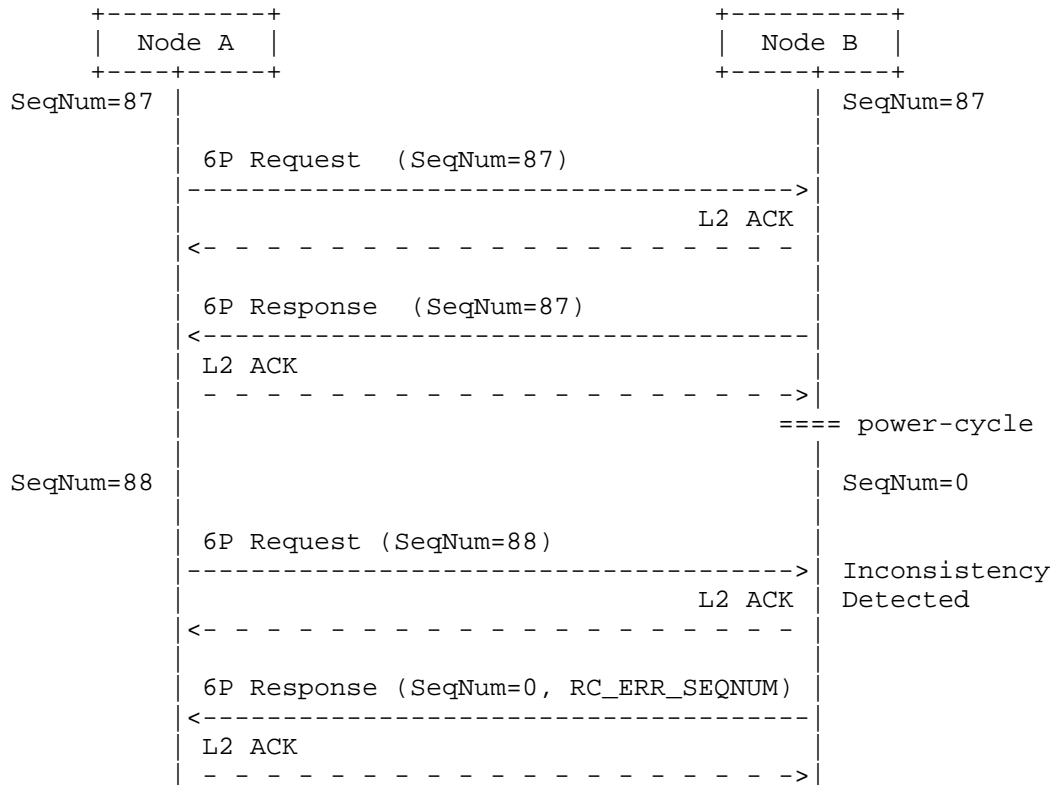


Figure 31: Example of inconsistency because of node B reset. Detected by node B

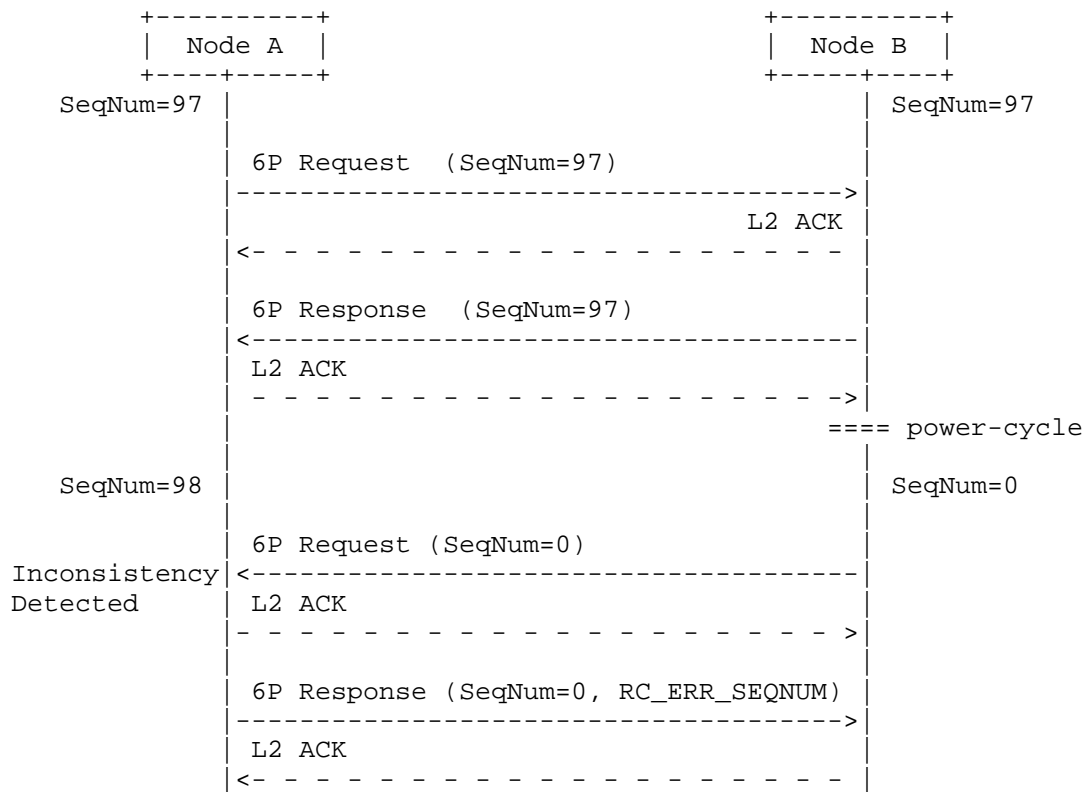


Figure 32: Example of inconsistency because node B resets. Detected by node A

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

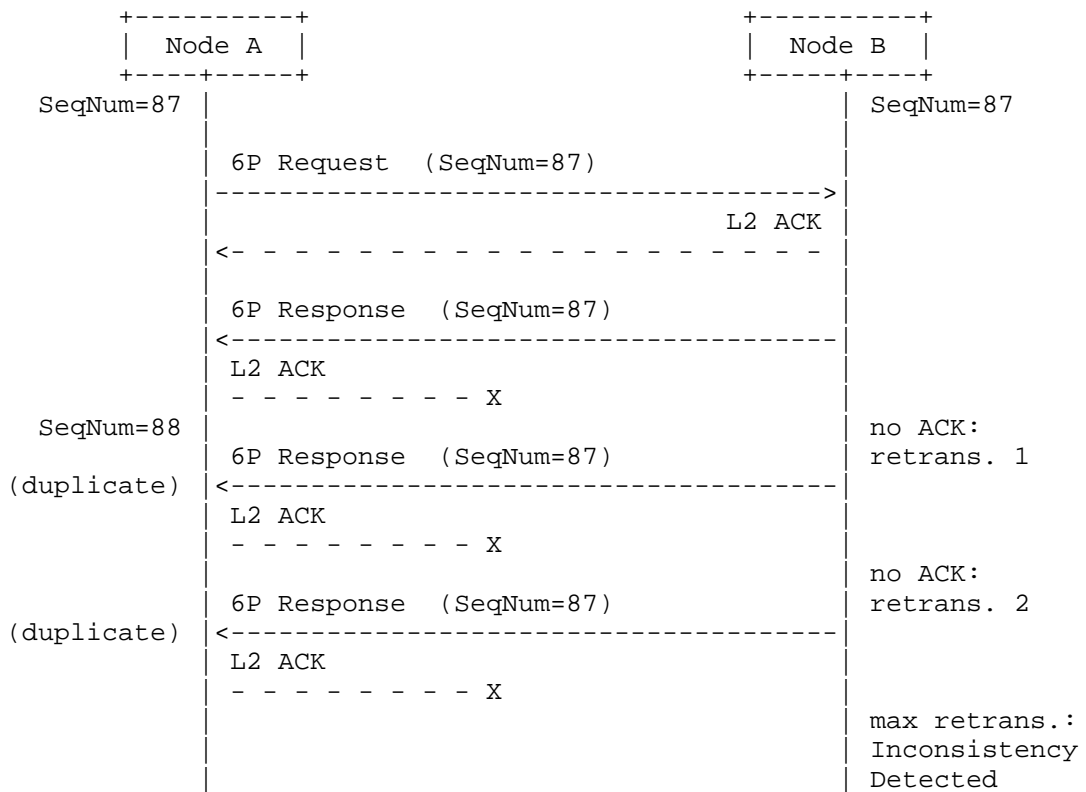


Figure 33: Example inconsistency because of maximum link-layer retransmissions (here 2).

In both cases, node B detects the inconsistency.

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

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

3.4.7. Handling Error Responses

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

3.5. Security

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

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

4.1. SF Identifier (SFID)

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

4.2. Requirements for an SF specification

The specification for an SF

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

- o MUST specify what to do after an error has occurred (either the node sent a 6P Response with an error code, or received one).
- o MUST specify the list of statistics to gather. Example statistics include the number of transmitted frames to each neighbor. In case the SF requires no statistics to be gathered, the specific of the SF MUST explicitly state so.

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

- o MAY redefine the format of the CellList field.
- o MAY redefine the format of the CellOptions field.
- o MAY redefine the meaning of the CellOptions field.

5. Security Considerations

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

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

6. IANA Considerations

6.1. IETF IE Subtype '6P'

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

Value	Subtype ID	Reference
<TBD>	SUBID_6TOP	RFCXXXX

Figure 34: IETF IE Subtype SUBID_6TOP.

6.2. 6TiSCH parameters sub-registries

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

6.2.1. 6P Version Numbers

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

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

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

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

Version	Reference
0	RFCXXXX

Figure 35: 6P Version Numbers.

All other Version Numbers are Unassigned.

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

6.2.2. 6P Message Types

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

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

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

Initial entries in this sub-registry are as follows:

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

Figure 36: 6P Message Types.

All other Message Types are Reserved.

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

6.2.3. 6P Command Identifiers

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

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

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

Initial entries in this sub-registry are as follows:

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

Figure 37: 6P Command Identifiers.

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

6.2.4. 6P Return Codes

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

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

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

Initial entries in this sub-registry are as follows:

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

Figure 38: 6P Return Codes.

All other Message Types are Unassigned.

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

6.2.5. 6P Scheduling Function Identifiers

6P Scheduling Function Identifiers.

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

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

Initial entries in this sub-registry are as follows:

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

Figure 39: SF Identifiers (SFID).

All other Message Types are Unassigned.

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

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

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

6.2.6. 6P CellOptions bitmap

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

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

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

Initial entries in this sub-registry are as follows:

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

Figure 41: 6P CellOptions bitmap.

All other Message Types are Reserved.

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

7. References

7.1. Normative References

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

7.2. Informative References

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

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

Appendix A. Recommended Structure of an SF Specification

The following section structure for a SF document is RECOMMENDED:

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

Authors' Addresses

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

Email: wangqin@ies.ustb.edu.cn

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

Email: xvilajosana@uoc.edu

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

Email: thomas.watteyne@analog.com

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

D. Dujovne, Ed.
Universidad Diego Portales
LA. Grieco
Politecnico di Bari
MR. Palattella
Luxembourg Institute of Science and Technology (LIST)
N. Accettura
LAAS-CNRS
March 5, 2018

6TiSCH Experimental Scheduling Function (SFX)
draft-ietf-6tisch-6top-sfx-01

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

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

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Scheduling Function Identifier	3
3. Allocated and Used Cells	3
4. Overprovisioning	3
5. Scheduling Algorithm	4
5.1. SFX Triggering Events	4
5.2. SFX Cell Estimation Algorithm	4
5.3. SFX Allocation Policy	5
6. Rules for CellList	7
7. 6P Timeout Value	7
8. Meaning of Metadata Information	8
9. Node Behavior at Boot	8
10. Cell Type	8
11. SFX Statistics	8
12. Relocating Cells	8
13. Forced Cell Deletion Policy	9
14. 6P Error Handling	9
15. Experimental requirements	9
16. Security Considerations	10
17. IANA Considerations	10
17.1. SFX Scheduling Function Identifiers	10
18. Acknowledgments	10
19. References	11
19.1. Normative References	11
19.2. Informative References	11
Authors' Addresses	11

1. Introduction

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

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

1. It waits for a triggering event (Section 5.1).
2. It applies the Cell Estimation Algorithm (CEA) for a particular neighbor to determine how many cells are required to that neighbor (Section 5.2).
3. It applies the Allocation Policy to compare the number of required cells to the number of already scheduled cells, and determines the number of cells to add/delete (Section 5.3).

SFX addresses the requirements for a scheduling function listed in Section 5.2 from [I-D.ietf-6tisch-6top-protocol], and follows the recommended outline listed in Section 5.3 of [I-D.ietf-6tisch-6top-protocol]. This document follows the terminology defined in [I-D.ietf-6tisch-terminology].

2. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of SFX is IANA_6TISCH_SFID_SFX.

3. Allocated and Used Cells

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

4. Overprovisioning

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

5. Scheduling Algorithm

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

5.1. SFX Triggering Events

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

5.2. SFX Cell Estimation Algorithm

The Cell Estimation Algorithm takes into account the number of currently used cells to the neighbor. This allows the algorithm to estimate a new number of cells to be scheduled to the neighbor. As a consequence, the Cell Estimation Algorithm for SFX follows the steps described below:

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

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

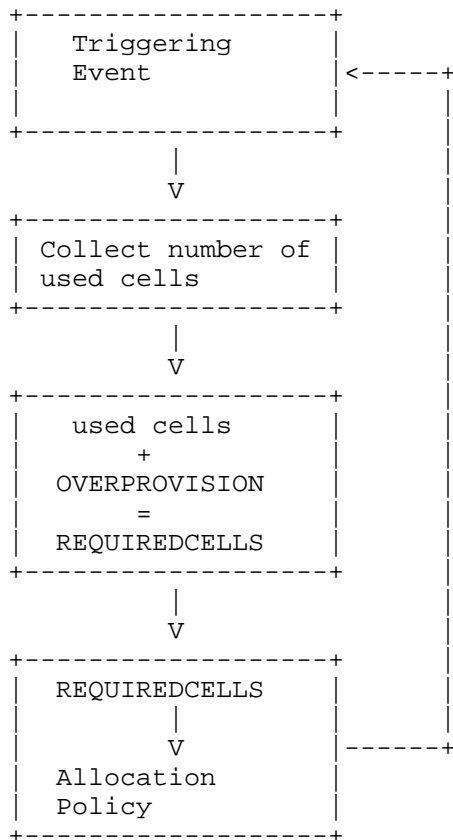


Figure 1: The SFX Estimation Algorithm

5.3. SFX Allocation Policy

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

SFX uses the following parameters:

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

specific. A setting of SFXTHRESH>0 causes the node to allocate at least SFXTHRESH cells to each of its' neighbors.

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

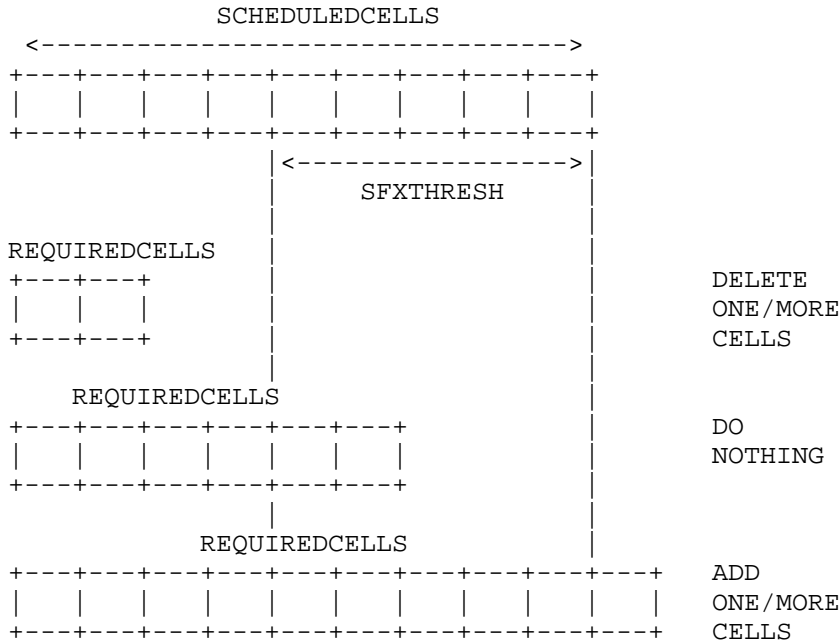


Figure 2: The SFX Allocation Policy

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

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

6. Rules for CellList

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

Whitelist case:

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

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

Blacklist case:

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

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

7. 6P Timeout Value

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

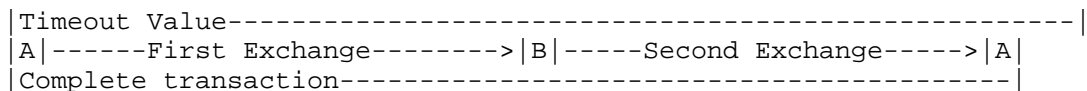


Figure 3: Example Transaction where the timeout does not expire

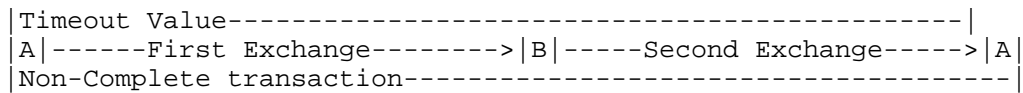


Figure 4: Example Transaction where the timeout expires

8. Meaning of Metadata Information

The Metadata 16-bit field is used as follows:

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

9. Node Behavior at Boot

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

10. Cell Type

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

11. SFX Statistics

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

12. Relocating Cells

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

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

13. Forced Cell Deletion Policy

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

14. 6P Error Handling

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

RC_SUCCESS:

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

RC_EOL: If an LIST command is issued and the RC_EOL is received, the node MUST understand what is specified on Section 3.3.5 of [I-D.ietf-6tisch-6top-protocol].

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

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

RC_ERR_SEQNUM: The node MUST issue a CLEAR command to the neighbor.

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

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

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

RC_RESET: Abort 6P Transaction.

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

15. Experimental requirements

In order to evaluate the performance of this draft, we propose the following experimental work:

1. Define values for OVERPROVISION, SFXTHRESH and ranges to the number of cells to Add or Delete after the Allocation Policy is applied for typical use cases.
2. Analyze the scheduling stability (in terms of oscillation) and the hysteresis effect on scheduling using SFX. A tradeoff shall be found between the reactivity of the algorithm facing new scheduling requirements and the number of overprovisioned cells.
3. Define the PDR value below the Average which is most effective for blacklisting cells and a method to whitelist cells. Analyze the stability and long-term behavior of this algorithm.
4. Measure the distribution of cell scheduling delay (including the time taken by 6P) to estimate timeouts for different type of transactions.

16. Security Considerations

SFX is defined as an algorithm designed to efficiently fulfill bandwidth requirements between neighbour nodes and does not define a new protocol SFX uses the Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration standardized on [RFC8180] and the 6top Protocol (6P): [I-D.ietf-6tisch-6top-protocol]. SFX relies on the security framework described on [I-D.ietf-6tisch-minimal-security].

17. IANA Considerations

17.1. SFX Scheduling Function Identifiers

This document provide a new element to the "6P Scheduling Function Identifiers" sub-registry, which is part of the "IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) parameters" registry, as defined by [I-D.ietf-6tisch-6top-protocol]. This Subtype is defined on figure Figure 5

SFID	Name	Reference
IANA_6TISCH_SFID_SFX	Experimental Scheduling Function (SFX)	RFCXXXX (NOTE:this)

Figure 5: IETF IE Subtype '6P'

18. Acknowledgments

Thanks to Kris Pister for his contribution in designing the default Bandwidth Estimation Algorithm. Thanks to Qin Wang and Thomas

Watteyne for their support in defining the interaction between SFX and the 6top sublayer.

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

19. References

19.1. Normative References

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

19.2. Informative References

- [I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-09 (work in progress), October 2017.
- [I-D.ietf-6tisch-minimal-security]
Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-04 (work in progress), October 2017.
- [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.

Authors' Addresses

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

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

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

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

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

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

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

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

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

M. Vucinic, Ed.
University of Montenegro
J. Simon
Analog Devices
K. Pister
University of California Berkeley
M. Richardson
Sandelman Software Works
March 05, 2018

Minimal Security Framework for 6TiSCH
draft-ietf-6tisch-minimal-security-05

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 a short link-layer address. This specification defines the message format, a new Stateless-Proxy CoAP option, and configures 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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Identifiers	4
4. One-Touch Assumption	5
5. Join Overview	5
5.1. Step 1 - Enhanced Beacon	7
5.2. Step 2 - Neighbor Discovery	7
5.3. Step 3 - Join Request	8
5.4. Step 4 - Join Response	8
6. Link-layer Configuration	9
7. Network-layer Configuration	9
7.1. Identification of Join Request Traffic	10
7.2. Identification of Join Response Traffic	11
8. Application-level Configuration	11
8.1. OSCORE Security Context	12
9. 6TiSCH Join Protocol	13
9.1. Specification of the Join Request	14
9.2. Specification of the Join Response	15
9.3. Error Handling and Retransmission	17
9.4. Rekeying and Rejoining	18
9.5. Parameters	18
9.6. Mandatory to Implement Algorithms	18
10. Stateless-Proxy CoAP Option	19
11. Security Considerations	20
12. Privacy Considerations	21
13. IANA Considerations	21
13.1. CoAP Option Numbers Registry	21
14. Acknowledgments	22
15. References	22
15.1. Normative References	22
15.2. Informative References	23

Appendix A. Example 24
Authors' Addresses 27

1. Introduction

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 for join purposes 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 [I-D.ietf-core-object-security] for its end-to-end security. The secure join solution defined in this document therefore reuses those two protocols as its building blocks.

This document defines a secure join solution for a new device, called "pledge", to securely join a 6TiSCH network. The specification defines a 6TiSCH Join Protocol (6JP) used by the pledge to request admission into a network managed by the JRC, and for the JRC to configure the pledge with the necessary parameters, a new CoAP option, and configures different layers of the 6TiSCH protocol stack for the join process to occur in a secure manner.

It assumes the presence of a JRC (join registrar/coordinator), a central entity. It further assumes that the pledge and the JRC share a symmetric key, called PSK (pre-shared key). The PSK is used to configure OSCORE to provide a secure channel to 6JP. How the PSK is installed is out of scope of this document.

When the pledge seeks admission to a 6TiSCH network, it first synchronizes to it, by initiating the passive scan defined in [IEEE802.15.4-2015]. The pledge then exchanges messages with the JRC; these messages can be forwarded by nodes already part of the 6TiSCH network. The messages exchanged allow the JRC and the pledge to mutually authenticate, based on the PSK. They also allow the JRC to configure the pledge with link-layer keying material and a short link-layer address. After this secure join process successfully completes, the joined node can interact with its neighbors to request additional bandwidth using the 6top Protocol [I-D.ietf-6tisch-6top-protocol] and start sending the application traffic.

2. Terminology

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

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

The specification also includes a set of informative examples using the CBOR diagnostic notation [I-D.ietf-cbor-cddl].

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

- o pledge
- o joined node
- o join proxy (JP)
- o join registrar/coordinator (JRC)
- o enhanced beacon (EB)
- o join protocol
- o join process

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

3. Identifiers

The "network identifier" uniquely identifies the 6TiSCH network in the namespace managed by a JRC. Typically, this is the 16-bit Personal Area Network Identifier (PAN ID) defined in [IEEE802.15.4-2015]. Companion documents can specify the use of a different network identifier for join purposes, but this is out of scope of this specification. Such identifier needs to be carried within Enhanced Beacon (EB) frames.

The "pledge identifier" uniquely identifies the pledge in the namespace managed by a JRC. The pledge identifier is typically the globally unique 64-bit Extended Unique Identifier (EUI-64) of the

IEEE 802.15.4 device. This identifier is used to generate the IPv6 addresses of the pledge and to identify it during the execution of the join protocol. For privacy reasons, it is possible to use an identifier different from the EUI-64 (e.g. a random string). See Section 12.

4. One-Touch Assumption

This document assumes a one-touch scenario. The 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 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 pledge, and that the two devices can exchange the parameters.

Parameters that are provisioned to the pledge include:

- o Pre-Shared Key (PSK). The JRC additionally needs to store the identifier of the pledge bound to the given PSK. The PSK SHOULD be at least 128 bits in length, generated uniformly at random. It is RECOMMENDED to generate the PSK with a cryptographically secure pseudorandom number generator. Each pledge SHOULD be provisioned with a unique PSK.
- o Optionally, a network identifier. Provisioning the network identifier to the pledge is RECOMMENDED, as it significantly speeds up the join process. In case this parameter is not provisioned, the pledge attempts to join one network at a time.
- o Optionally, any non-default algorithms. Mandatory to implement and default algorithms are specified in Section 9.6.

5. Join 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 [IEEE802.15.4-2015]. This frame provides network synchronization information, and tells the device when it can send a frame to the node sending the beacons, which plays the role of join proxy (JP) for the pledge, and when it can expect to receive a frame.
2. The pledge configures its link-local IPv6 address and advertises it to the join proxy (JP).
3. The pledge sends a Join Request to the JP in order to securely identify itself to the network. The Join Request is directed to the JRC, which may be co-located on the JP or another device.
4. In case of successful processing of the request, the pledge receives a join response from JRC (via the JP) that sets up one or more link-layer keys used to authenticate and encrypt subsequent transmissions to peers, and a short link-layer address for the pledge.

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:

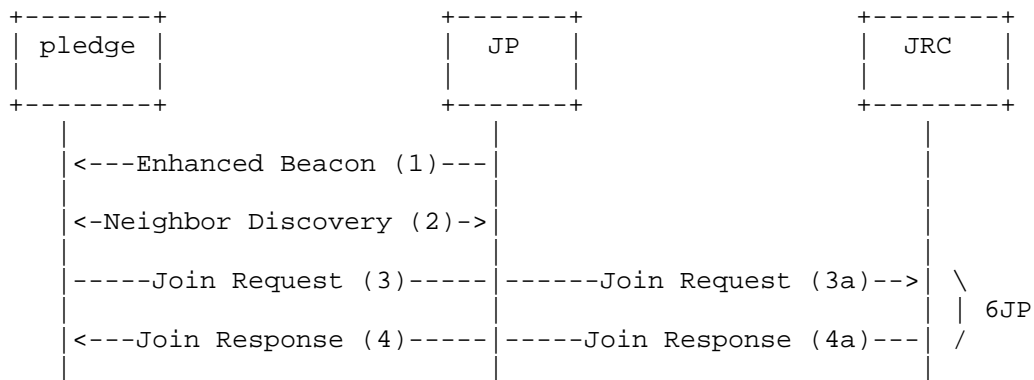


Figure 1: Overview of a successful join process. 6JP stands for 6TiSCH Join Protocol.

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

5.1. 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 [IEEE802.15.4-2015], 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, 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 9.3.

Once the pledge selects the EB, it synchronizes to it and transitions into a low-power mode. It deeply duty cycles its radio, switching the radio on when the provided schedule indicates slots which 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 plays the role of JP.

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

5.2. 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 5.5.1 of [RFC6775]. 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 JP accepts these unprotected frames is discussed in Section 6.

5.3. Step 3 - Join Request

The Join Request is a message sent from the pledge to the JP, and which the JP forwards to the JRC. The JP forwards the Join Request to the JRC on the existing 6TiSCH network. How exactly this happens is out of scope of this document; some networks may wish to dedicate specific slots for this join traffic.

The Join Request is authenticated/encrypted end-to-end using an AEAD (Authenticated Encryption with Associated Data) algorithm from [RFC8152] and a key derived from the PSK, the pledge identifier and a request-specific constant value. Algorithms which MUST be implemented are specified in Section 9.6.

The nonce used when securing the Join Request is derived from the PSK, the pledge identifier and a monotonically increasing counter initialized to 0 when first starting.

Join Request message is specified in Section 9.1, while the details on security processing can be found in Section 7 of [I-D.ietf-core-object-security].

5.4. Step 4 - Join Response

The Join Response is sent by the JRC to the pledge, and is forwarded through the JP as it serves as a stateless relay. The packet containing the Join Response travels from the JRC to JP using the operating routes in the 6TiSCH network. The JP delivers it to the pledge. The JP operates as the application-layer proxy, and does not keep any state to relay the message. It uses information sent in the clear within the Join Response to decide where to forward to.

The Join Response is authenticated/encrypted end-to-end using an AEAD algorithm from [RFC8152]. The key used to protect the response is different from the one used to protect the request (both are derived from the PSK, as explained in Section 8.1). The response is protected using the same nonce as in the request.

The Join Response contains one or more link-layer key(s) that the pledge will use for subsequent communication. Each key that is provided by the JRC is associated with an 802.15.4 key identifier. In other link-layer technologies, a different identifier may be substituted. The Join Response also contains an IEEE 802.15.4 short address [IEEE802.15.4-2015] assigned by the JRC to the pledge, and optionally the IPv6 address of the JRC.

Join Response message is specified in Section 9.2, while the details on security processing can be found in Section 7 of [I-D.ietf-core-object-security].

6. Link-layer Configuration

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

As specified in [RFC8180], the network uses a key termed as K1 to authenticate EBs and a key termed as K2 to authenticate and optionally encrypt DATA and ACKNOWLEDGMENT frames. The keys K1 and K2 MAY be the same key (same value and IEEE 802.15.4 index). How the JRC communicates these keys to 6LBR is out of scope of this specification.

The pledge does not initially do any authenticity check of the EB frames, as it does not know the K1 key. 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. The JP accepts these frames (using the "exempt mode" in 802.15.4) for the duration of the join process. How the JP learns whether the join process is ongoing is out of scope of this specification.

As the EB itself cannot be authenticated by the pledge, an attacker may craft a frame that appears to be a valid EB, since the pledge can neither know the ASN a priori nor verify the address of the JP. This opens up a possibility of DoS attack, as discussed in Section 11. Beacon authentication keys are discussed in [RFC8180].

7. 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. How the pledge and the JP decide to transition each other from untrusted to trusted cache, once the join process completes, is out of scope. One implementation technique is to use the information whether the incoming frames are secured at the link layer.

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, and with the JRC at the application layer, as specified in Section 8.

The JP communicates with the JRC over global IPv6 addresses. The JP discovers the network 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 actual IPv6 address of the JRC from the Join Response, as specified in Section 9.2; it uses it once joined in order to operate as a JP.

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 RPL DIOs [RFC6550] to its IPv6 address. The pledge learns the address of the JRC once joined and upon the reception of a first RPL DIO message, and uses it to operate as a JP.

Before the 6TiSCH network is started, the 6LBR MUST be provisioned with the IPv6 address of the JRC.

7.1. Identification of Join Request Traffic

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

When operating as part of a [RFC8180] 6TiSCH minimal network using reasonable scheduling algorithms, the join request traffic present 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 is join request traffic, and so can avoid allocating additional bandwidth itself. The Join Proxy SHOULD implement a bandwidth cap on outgoing join request traffic. This cap will not protect intermediate nodes as they can not tell join request traffic from regular traffic. Despite the bandwidth cap implemented separately on each Join Proxy, the aggregate join request 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 join request traffic. In order to permit the intermediate nodes to avoid this, the traffic needs to be tagged in some way.

[RFC2597] defines a set of per-hop behaviors that may be encoded into the Diffserv Code Points (DSCPs). The Join Proxy SHOULD set the DSCP

of join request packets that it produces as part of the relay process to AF43 code point (See Section 6 of [RFC2597]).

A Join Proxy that does not set the DSCP 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.

7.2. Identification of Join Response Traffic

The JRC SHOULD set the DSCP of join response packets addressed to the Join Proxy to AF42 code point. Join response traffic can not be induced by an attacker as it is generated only in response to legitimate pledges (see Section 9.3). AF42 has lower drop probability than AF43, giving join response traffic priority in buffers over join request traffic.

When the JRC is not co-located with the 6LBR, then the code point provides a clear indication to the 6LBR that this is join response traffic.

Due to the convergecast nature of the DODAG, the 6LBR links are often the most congested, 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.

8. Application-level Configuration

The Join Request/Join Response exchange in Figure 1 is carried over CoAP [RFC7252] and secured using OSCORE [I-D.ietf-core-object-security]. The pledge plays the role of a CoAP client; the JRC plays the role of 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. Rather, the JP processes forwarding-related CoAP options and makes requests on behalf of the pledge, in a stateless manner by using the Stateless-Proxy option defined in this document.

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 9.1.

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.

The JP MUST add a Stateless-Proxy option to all the requests that it forwards on behalf of the pledge as part of the join process.

The value of the Stateless-Proxy option is set to the internal JP state, needed to forward the Join Response message to the pledge. The Stateless-Proxy option handling is defined in Section 10.

The JP also tags all packets carrying the Join Request message at the network layer, as specified in Section 7.1.

8.1. OSCORE Security Context

Before the pledge and the JRC may start exchanging CoAP messages protected with OSCORE, they need to derive the OSCORE security context from the parameters provisioned out-of-band, as discussed in Section 4.

The OSCORE security context MUST be derived at the pledge and the JRC as per Section 3 of [I-D.ietf-core-object-security].

- o the Master Secret MUST be the PSK.
- o the Master Salt MUST be the pledge identifier.
- o the Sender ID of the pledge MUST be set to byte string 0x00.
- o the Recipient ID (ID of the JRC) MUST be set to byte string 0x01.
- o the Algorithm MUST be set to the value from [RFC8152], agreed out-of-band by the same mechanism used to provision the PSK. The default is AES-CCM-16-64-128.
- o the Key Derivation Function MUST be agreed out-of-band. Default is HKDF SHA-256 [RFC5869].

The derivation in [I-D.ietf-core-object-security] results in traffic 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

and sender identifier. For details on nonce construction, refer to [I-D.ietf-core-object-security].

Implementations MUST ensure that multiple CoAP requests to different JRCs result in the use of the same OSCORE context, so that the sequence numbers are properly incremented for each request. 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. A simple implementation technique is to instantiate the OSCORE security context with a given PSK only once and use it for all subsequent requests. Failure to comply will break the confidentiality property of the AEAD algorithm due to the nonce reuse.

8.1.1. Persistency

Implementations MUST ensure that mutable OSCORE context parameters (Sender Sequence Number, Replay Window) are stored in persistent memory. A technique that prevents reuse of sequence numbers, detailed in Section 6.5.1 of [I-D.ietf-core-object-security], 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 pledge and the JRC is rare, making security outweigh the cost of writing to persistent memory.

9. 6TiSCH Join Protocol

6TiSCH Join Protocol (6JP) is a lightweight protocol over CoAP [RFC7252] and a secure channel provided by OSCORE [I-D.ietf-core-object-security]. 6JP allows the pledge to request admission into a network managed by the JRC, and for the JRC to configure the pledge with the parameters necessary for joining the network. These parameters are: link-layer keys in use, IEEE 802.15.4 short address assigned to the pledge, and the IPv6 address of the JRC.

This section specifies the 6JP bindings to CoAP and OSCORE, 6JP message formats and the semantics of different fields.

6JP 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.

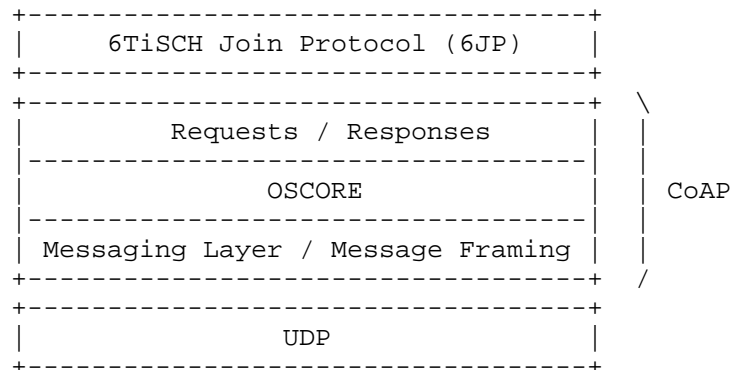


Figure 2: Abstract layering of 6JP.

6JP consists of two messages:

- o the Join Request message, sent by the pledge to the JRC, proxied by the JP. The Join Request message and its mapping to CoAP is specified in Section 9.1.
- o the Join Response message, sent by the JRC to the 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 pledge is authorized to join the network. The Join Response message is proxied by the JP. The Join Response message and its mapping to CoAP is specified in Section 9.2.

The payload of 6JP messages is encoded with CBOR [RFC7049], with some parameters being optional. The first byte of the CBOR-encoded byte string contains the CBOR major type and additional information (e.g. the number of elements in an array). In case of an array, the CBOR decoder decides based on this additional information if a certain optional parameter is present or not.

9.1. Specification of the Join Request

The Join Request the pledge sends SHALL be mapped to a CoAP request:

- o The request method is POST.
- o The type is Non-confirmable (NON).
- o The Proxy-Scheme option is set to "coap".
- o The Uri-Host option is set to "6tisch.arpa".

- o The Uri-Path option is set to "j".
- o The Object-Security option SHALL be set according to [I-D.ietf-core-object-security]. The OSCORE security context used is the one derived in Section 8.1. The OSCORE Context Hint SHALL be set to the pledge identifier. The OSCORE Context Hint allows the JRC to retrieve the security context for a given pledge.
- o The payload is a CBOR array [RFC7049] containing, in order:
 - * Byte string, containing the identifier of the network that the pledge is attempting to join. This enables the JRC to manage multiple 6TiSCH networks.

```
request_payload = [  
    network_identifier : bstr,  
]
```

9.2. Specification of the Join Response

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

- o The response Code is 2.04 (Changed).
- o The payload is a CBOR array [RFC7049] containing, in order:
 - * the COSE Key Set, specified in [RFC8152], containing one or more link-layer keys. The mapping of individual keys to 802.15.4-specific parameters is described in Section 9.2.1.
 - * the link-layer short address to be used by the pledge. The format of the short address follows Section 9.2.2.
 - * optionally, the IPv6 address of the JRC, encoded as a byte string, with the length of 16 bytes. If the IPv6 address of the JRC is not present in the Join Response, this indicates the JRC is co-located with the 6LBR, and has the same IPv6 address as the 6LBR. See Section 7.

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

9.2.1. Link-layer Keys Transported in the COSE Key Set

Each key in the COSE Key Set [RFC8152] SHALL be a symmetric key. The first key in the COSE Key Set SHALL be used as the K1 key from [RFC8180]. The second key in the COSE Key Set SHALL be used as the K2 key from [RFC8180]. In the case where the network uses the same key for K1 and K2, the COSE Key Set SHALL carry a single key.

If the COSE Key Set carries more than 2 keys, the implementation SHOULD consider the response as malformed.

If the "kid" parameter of the COSE Key structure is present, the corresponding key SHALL be used as IEEE 802.15.4 KeyIdMode 0x01 (index). In that case, parameter "kid" of the COSE Key structure SHALL be used to carry the IEEE 802.15.4 KeyIndex value.

If the length of the "kid" parameter is more than 1 byte (length defined by [IEEE802.15.4-2015]), the implementation SHOULD consider the response as malformed.

If the "kid" parameter is not present in the transported key, the implementation SHALL consider the key to be an IEEE 802.15.4 KeyIdMode 0x00 (implicit) key.

This document does not support IEEE 802.15.4 KeyIdMode 0x02 and 0x03 class keys. In the case that the response is considered malformed, the implementation SHOULD indicate to the user through an out-of-band mechanism the presence of an error condition.

9.2.2. Short Address

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

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

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

It is up to the joined node to request a new short address before the expiry of its previous address. The mechanism by which the node requests renewal is the same as during join procedure, as described in Section 9.4.

9.3. Error Handling and Retransmission

Since the Join Request is mapped to a Non-confirmable CoAP message, OSCORE processing at the JRC will silently drop the request in case of a failure. This 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 the Join Request at the JRC prevents a DoS attack where an attacker could force the pledge to attempt joining one network at a time, until all networks have been tried.

Using a Non-confirmable CoAP message to transport the Join Request also helps minimize the required CoAP state at the pledge and the Join Proxy, keeping it to a minimum typically needed to perform CoAP congestion control. It does, however, introduce some complexity as the pledge needs to implement a retransmission mechanism.

The following binary exponential back-off algorithm is inspired by the one described in [RFC7252]. For each Join Request the pledge sends while waiting for a Join Response, the pledge MUST keep track of a timeout and a retransmission counter. For a new Join Request, the timeout is set to a random value between `TIMEOUT_BASE` and `(TIMEOUT_BASE * TIMEOUT_RANDOM_FACTOR)`, and the retransmission counter is set to 0. When the timeout is triggered and the retransmission counter is less than `MAX_RETRANSMIT`, the Join Request is retransmitted, the retransmission counter is incremented, and the timeout is doubled. Note that the retransmitted Join Request passes new OSCORE processing, such that the sequence number in the OSCORE context is properly incremented. If the retransmission counter reaches `MAX_RETRANSMIT` on a timeout, the pledge SHOULD attempt to join the next advertised 6TiSCH network. If the pledge receives a Join Response that successfully passes OSCORE processing, it cancels the pending timeout and processes the response. The pledge MUST silently discard any response not protected with OSCORE, including error codes. For default values of retransmission parameters, see Section 9.5.

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

9.4. Rekeying and Rejoining

This specification handles initial keying of the pledge. For reasons such as rejoining after a long sleep, expiry of the short address, or node-initiated rekeying, the joined node MAY send a new Join Request using the already-established OSCORE security context. The JRC then responds with up-to-date keys and a (possibly new) short address. How the joined node decides when to rekey is out of scope of this document. Mechanisms for rekeying the network are defined in companion specifications.

9.5. Parameters

6JP uses the following parameters:

Name	Default Value
TIMEOUT_BASE	10 s
TIMEOUT_RANDOM_FACTOR	1.5
MAX_RETRANSMIT	4

The values of `TIMEOUT_BASE`, `TIMEOUT_RANDOM_FACTOR`, `MAX_RETRANSMIT` 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.

9.6. 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 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.

10. Stateless-Proxy CoAP Option

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 host, and the UDP source port number. If the JP used the stateful CoAP proxy defined in [RFC7252], it would be prone to Denial-of-Service (DoS) attacks, due to its limited memory.

The Stateless-Proxy CoAP option Figure 3 allows the JP to be entirely stateless. This option inserts, in the request, the state information needed for relaying the response back to the client. The proxy still keeps some general state (e.g. for congestion control or request retransmission), but no per-client state.

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

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

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

Figure 3: Stateless-Proxy CoAP Option

Upon reception of a Stateless-Proxy option, the CoAP server MUST echo it in the response. The value of the Stateless-Proxy option is internal proxy state that is opaque to the server. Example state information includes the IPv6 address of the client, its UDP source port, and the CoAP token. For security reasons, the state information MUST be authenticated, MUST include a freshness indicator (e.g. a sequence number or timestamp) and MAY be encrypted. The proxy may use an appropriate COSE structure [RFC8152] to wrap the state information as the value of the Stateless-Proxy option. The key used for encryption/authentication of the state information may be known only to the proxy.

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

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

11. Security Considerations

This document recommends that the pledge and JRC are provisioned with unique PSKs. The nonce used for the Join Request and the Join Response is the same, but used under a different key. The design differentiates between keys derived for requests and keys derived for responses by different sender identifiers (0x00 for pledge and 0x01 for JRC). Note that the address of the JRC does not take part in nonce or key construction. Even in the case of a misconfiguration in which the same PSK is used for several pledges, the keys used to protect the requests/responses from/towards different pledges are different, as they are derived using the pledge identifier as Master Salt. The PSK is still important for mutual authentication of the pledge and the JRC. Should an attacker come to know the PSK, then a man-in-the-middle attack is possible. The well-known problem with Bluetooth headsets with a "0000" pin applies here.

Being a stateless relay, the JP blindly forwards the join traffic into the network. A simple bandwidth cap on the JP prevents it from forwarding more traffic than the network can handle. This forces attackers to use more than one Join Proxy 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 DoS attacks 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 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. Bogus beacons prolong the join time of the pledge, and so the time spent in "minimal" [RFC8180] duty cycle mode.

12. Privacy Considerations

The join solution specified in this document relies on the uniqueness of the pledge identifier within the namespace managed by the JRC. This identifier is transferred in clear as an OSCORE Context Hint. 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 protocol is executed rarely compared to the network lifetime, long-term threats that arise from using EUI-64 as the pledge identifier are minimal. In addition, the Join Response message contains a short address which is assigned by the JRC to the 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 uses the short addresses for all further layer 2 (and layer-3) operations. This mitigates the aforementioned privacy risks as the short layer-2 address (visible even when the network is encrypted) is not traceable between locations and does not disclose the manufacturer, as is the case of EUI-64.

13. IANA Considerations

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

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

13.1. CoAP Option Numbers Registry

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

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

14. Acknowledgments

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

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

15. References

15.1. Normative References

- [I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", draft-ietf-core-object-security-08 (work in progress), January 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [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>>.

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

15.2. Informative References

- [I-D.ietf-6tisch-6top-protocol]
Wang, Q., Vilajosana, X., and T. Watteyne, "6top Protocol (6P)", draft-ietf-6tisch-6top-protocol-09 (work in progress), October 2017.
- [I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-09 (work in progress), June 2017.
- [I-D.ietf-cbor-cddl]
Birkholz, H., Vignano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-ietf-cbor-cddl-02 (work in progress), February 2018.
- [I-D.richardson-6tisch-minimal-rekey]
Richardson, M., "Minimal Security rekeying mechanism for 6TiSCH", draft-richardson-6tisch-minimal-rekey-02 (work in progress), August 2017.
- [IEEE802.15.4-2015]
IEEE standard for Information Technology, ., "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015.
- [RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <<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>>.
- [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>>.

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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. Example

Figure 4 illustrates a successful join protocol exchange. The pledge instantiates the OSCORE context and derives the traffic 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 Context Hint. Triggered by the presence of a Proxy-Scheme option, the JP forwards the request to the JRC and adds the Stateless-Proxy option with value set to the internally needed state, authentication tag, and a freshness indicator. 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 Context Hint parameter. It reconstructs OSCORE's external Additional Authenticated Data (AAD) needed for verification based on:

- o the Version of the received CoAP header.

- o the Algorithm value agreed out-of-band, default being AES-CCM-16-64-128 from [RFC8152].
- o the Request ID being set to the value of the "kid" field of the received COSE object.
- o the Join Request sequence number set to the value of "Partial IV" field of the received COSE object.
- o Integrity-protected options received as part of the request.

Replay protection is ensured by OSCORE and through persistent handling of mutable context parameters. Once the JP receives the Join Response, it authenticates the Stateless-Proxy option before deciding where to forward. The JP sets its internal state to that found in the Stateless-Proxy option, and forwards the Join Response to the correct pledge. Note that the JP does not possess the key to decrypt the COSE object (join_response) 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.

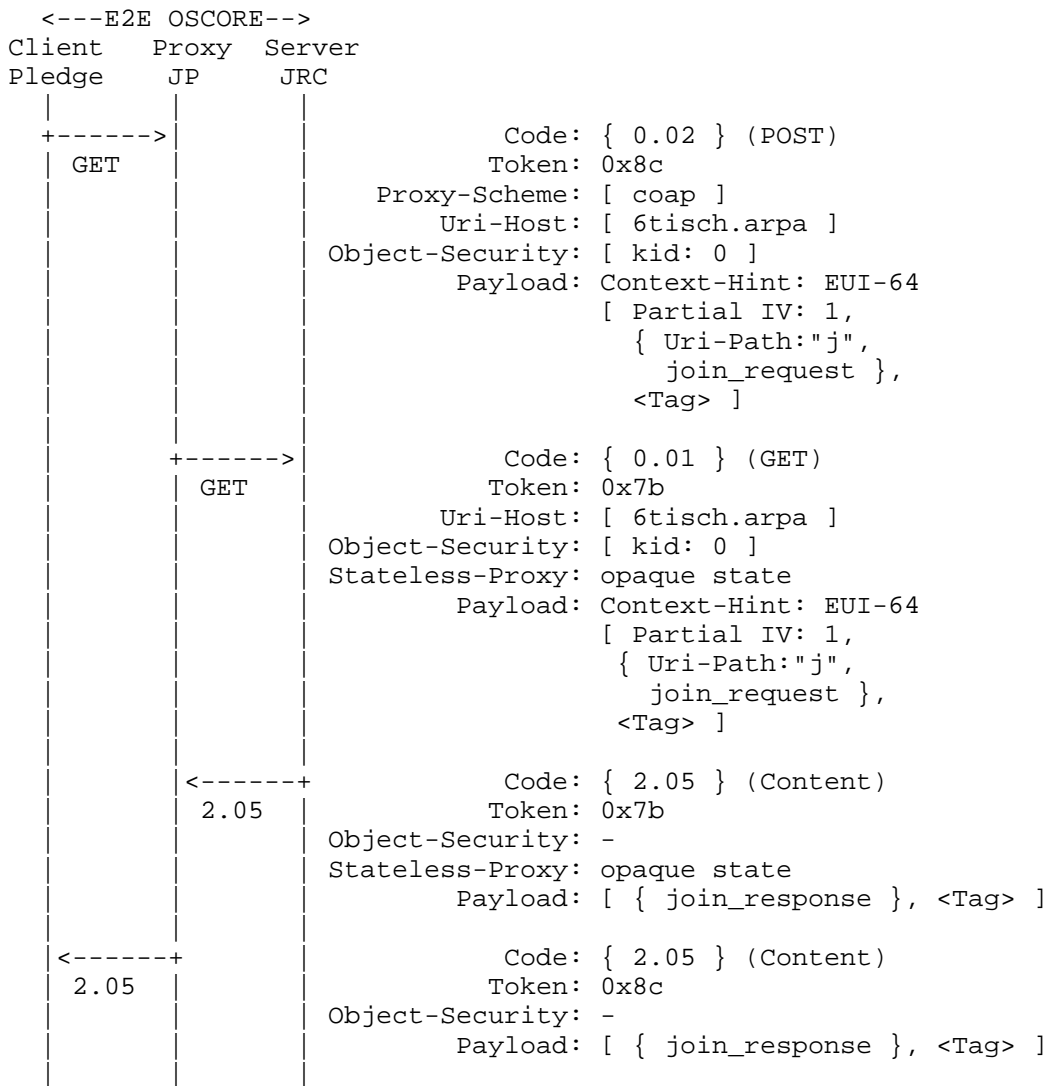


Figure 4: Example of a successful join protocol exchange. { ... } denotes encryption and authentication, [...] denotes authentication.

Where join_request is:

```
join_request:
[
  h'cafe' / PAN ID of the network pledge is attempting to join /
]
```

The `join_request` encodes to `h'8142cafe'` with a size of 4 bytes.

And `join_response` is:

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

The `join_response` encodes to `h'8281a301040241012050e6bf4287c2d7618d6a9687445ffd33e68142af93'` with a size of 30 bytes.

Authors' Addresses

Malisa Vucinic (editor)
University of Montenegro
Dzordza Vasingtona bb
Podgorica 81000
Montenegro

Email: malisav@ac.me

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

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

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

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

1. Introduction

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

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

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

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

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

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

2. Terminology

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

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

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

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

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

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

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

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

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

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

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

3. Provisioning Phase

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

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

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

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

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

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

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

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

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

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

4. Join Process Overview

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

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

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

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

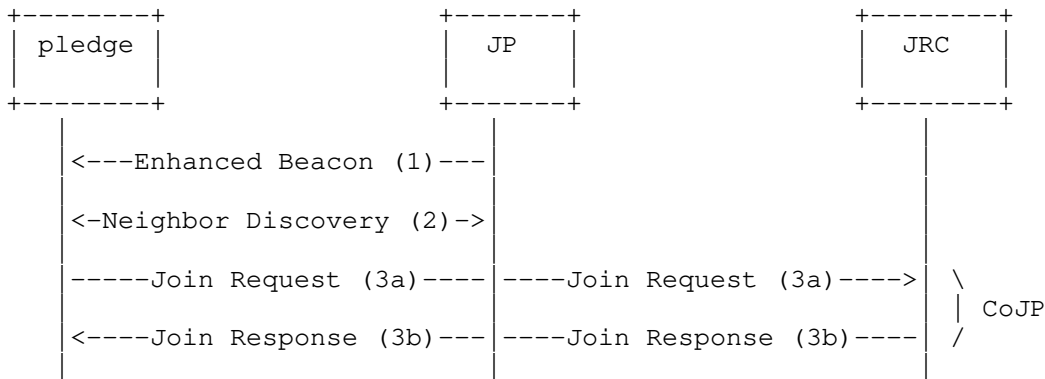


Figure 1: Overview of a successful join process.

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

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

4.1. Step 1 - Enhanced Beacon

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

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

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

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

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

4.2. Step 2 - Neighbor Discovery

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

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

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

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

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

4.3.1. Step 3a - Join Request

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

4.3.2. Step 3b - Join Response

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

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

4.4. The Special Case of the 6LBR Pledge Joining

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

5. Link-layer Configuration

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

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

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

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

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

5.1. Distribution of Time

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

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

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

6. Network-layer Configuration

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

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

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

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

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

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

6.1. Identification of Unauthenticated Traffic

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

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

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

6.1.1.1. Traffic from JP to JRC

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

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

6.1.1.2. Traffic from JRC to JP

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

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

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

7. Application-level Configuration

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

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

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

7.1. Statelessness of the JP

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

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

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

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

7.2. Recommended Settings

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

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

Recommended CoAP settings.

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

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

7.3. OSCORE

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

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

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

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

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

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

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

7.3.1. Replay Window and Persistency

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

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

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

7.3.2. OSCORE Error Handling

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

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

7.3.3. Mandatory to Implement Algorithms

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

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

8. Constrained Join Protocol (CoJP)

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

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

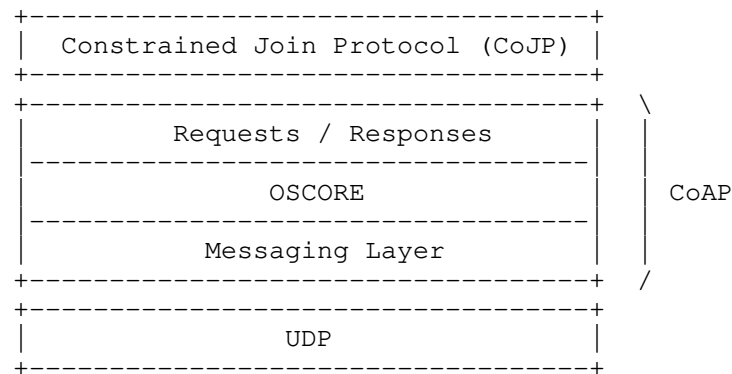


Figure 2: Abstract layering of CoJP.

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

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

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

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

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

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

8.1. Join Exchange

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

8.1.1. Join Request Message

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

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

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

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

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

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

8.1.2. Join Response Message

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

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

8.2. Parameter Update Exchange

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

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

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

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

8.2.1. Parameter Update Message

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

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

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

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

8.3. Error Handling

8.3.1. CoJP CBOR Object Processing

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

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

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

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

8.3.2. Diagnostic Response Message

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

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

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

8.3.3. Failure Handling

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

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

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

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

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

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

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

8.4. CoJP Objects

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

8.4.1. Join Request Object

The `Join_Request` structure is built on a CBOR map object.

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

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

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

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

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

Table 1: Summary of `Join_Request` parameters.

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

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

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

Table 2: Role values.

8.4.2. Configuration Object

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

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

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

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

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

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

Table 3: Summary of Configuration parameters.

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

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

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

Table 4: CoJP parameters map labels.

8.4.3. Link-Layer Key

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

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

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

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

- o `key_id`: The identifier of the key, encoded as a CBOR unsigned integer. This parameter **MUST** be included. If the decoded CBOR unsigned integer value is larger than the maximum link-layer key identifier, the key is considered invalid. In case the key is considered invalid, the key **MUST** be discarded and the implementation **MUST** signal the error as specified in Section 8.3.1.
- o `key_usage`: The identifier of the link-layer algorithm, security level and link-layer frame types that can be used with the key, encoded as an integer. This parameter **MAY** be included. Possible values and the corresponding link-layer settings are specified in IANA "CoJP Key Usage" registry (Section 11.2). In case the parameter is omitted, the default value of 0 (6TiSCH-K1K2-ENC-MIC32) from Table 5 **MUST** be assumed. This default value has been chosen such that it results in byte savings in the most constrained settings but does not imply a recommendation for its general usage.
- o `key_value`: The value of the cryptographic key, encoded as a byte string. This parameter **MUST** be included. If the length of the byte string is different than the corresponding key length for a given algorithm specified by the `key_usage` parameter, the key **MUST** be discarded and the implementation **MUST** signal the error as specified in Section 8.3.1.
- o `key_addinfo`: Additional information needed to configure the link-layer key, encoded as a byte string. This parameter **MAY** be included. The processing of this parameter is dependent on the link-layer technology in use and a particular keying mode.

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

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

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

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

K1K2-MIC128			CCM-128	for EBs, DATA and AC KNOWLEDGMEN T.	ocument]]
6TiSCH-K1-MIC32	6		IEEE802154-AES- CCM-128	Use MIC-32 for EBs.	[[this d ocument]]
6TiSCH-K1-MIC64	7		IEEE802154-AES- CCM-128	Use MIC-64 for EBs.	[[this d ocument]]
6TiSCH-K1-MIC12 8	8		IEEE802154-AES- CCM-128	Use MIC-128 for EBs.	[[this d ocument]]
6TiSCH-K2-MIC32	9		IEEE802154-AES- CCM-128	Use MIC-32 for DATA and ACKNOWL EDGMENT.	[[this d ocument]]
6TiSCH-K2-MIC64	10		IEEE802154-AES- CCM-128	Use MIC-64 for DATA and ACKNOWL EDGMENT.	[[this d ocument]]
6TiSCH-K2-MIC12 8	11		IEEE802154-AES- CCM-128	Use MIC-128 for DATA and ACKNOWL EDGMENT.	[[this d ocument]]
6TiSCH-K2-ENC- MIC32	12		IEEE802154-AES- CCM-128	Use ENC- MIC-32 for DATA and AC KNOWLEDGMEN T.	[[this d ocument]]
6TiSCH-K2-ENC- MIC64	13		IEEE802154-AES- CCM-128	Use ENC- MIC-64 for DATA and AC KNOWLEDGMEN T.	[[this d ocument]]
6TiSCH-K2-ENC- MIC128	14		IEEE802154-AES- CCM-128	Use ENC- MIC-128 for DATA and AC KNOWLEDGMEN	[[this d ocument]]

			T.
--	--	--	----

Table 5: Key Usage values.

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

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

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

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

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

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

8.4.3.3. Use in IEEE Std 802.15.4

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

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

- o Key ID Mode 0x00 (Implicit, pairwise): key_id parameter MUST be set to 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to the link-layer address(es) of a single peer with whom the key should be used. Depending on the configuration of the network, key_addinfo may carry the peer's long link-layer address (i.e. pledge identifier), short link-layer address, or their concatenation with the long address being encoded first. Which address type(s) is carried is determined from the length of the byte string.
- o Key ID Mode 0x01 (Key Index): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST NOT be present.
- o Key ID Mode 0x02 (4-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 4 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].
- o Key ID Mode 0x03 (8-byte Explicit Key Source): key_id parameter MUST be set to a value different than 0. key_addinfo parameter MUST be present. key_addinfo parameter MUST be set to a byte string, exactly 8 bytes long. key_addinfo parameter carries the Key Source parameter used to configure [IEEE802.15.4].

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

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

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

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

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

8.4.4. Short Identifier

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

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

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

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

8.4.4.1. Use in IEEE Std 802.15.4

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

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

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

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

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

8.4.5. Unsupported Configuration Object

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

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

- o `code`: Indicates the capability of acting on the parameter signaled by `parameter_label`, encoded as an integer. This parameter **MUST** be included. Possible values of this parameter are specified in the IANA "CoJP Unsupported Configuration Code Registry" (Section 11.3).
- o `parameter_label`: Indicates the parameter. This parameter **MUST** be included. Possible values of this parameter are specified in the label column of the IANA "CoJP Parameters" registry (Section 11.1).
- o `parameter_addinfo`: Additional information about the parameter that cannot be acted upon. This parameter **MUST** be included. In case the code is set to "Unsupported", `parameter_addinfo` gives additional information to the JRC. If the parameter indicated by `parameter_label` cannot be acted upon regardless of its value, `parameter_addinfo` **MUST** be set to null, signaling to the JRC that it **SHOULD NOT** attempt to configure the parameter again. If the pledge can act on the parameter, but cannot configure the setting indicated by the parameter value, the pledge can hint this to the JRC. In this case, `parameter_addinfo` **MUST** be set to the value of the parameter that cannot be acted upon following the normative parameter structure specified in this document. For example, it is possible to include the link-layer key set object, signaling a subset of keys that cannot be acted upon, or the entire key set that was received. In that case, the value of the `parameter_addinfo` follows the link-layer key set structure defined in Section 8.4.2. In case the code is set to "Malformed", `parameter_addinfo` **MUST** be set to null, signaling to the JRC that it **SHOULD NOT** attempt to configure the parameter again.

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

```
Unsupported_Configuration = [  
    + parameter           : Unsupported_Parameter  
]  
  
Unsupported_Parameter = (  
    code                 : int,  
    parameter_label     : int,  
    parameter_addinfo   : nil / any  
)
```

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

Table 6: Unsupported Configuration code values.

8.5. Recommended Settings

This section gives RECOMMENDED values of CoJP settings.

Name	Default Value
COJP_MAX_JOIN_ATTEMPTS	4
COJP_REKEYING_GUARD_TIME	12 seconds

Recommended CoJP settings.

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

9. Security Considerations

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

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

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

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

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

The JP forwards the unauthenticated join traffic into the network. A data cap on the JP prevents it from forwarding more traffic than the network can handle and enables throttling in case of an attack. Note that this traffic can only be directed at the JRC so that the JRC needs to be prepared to handle such unsanitised 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.

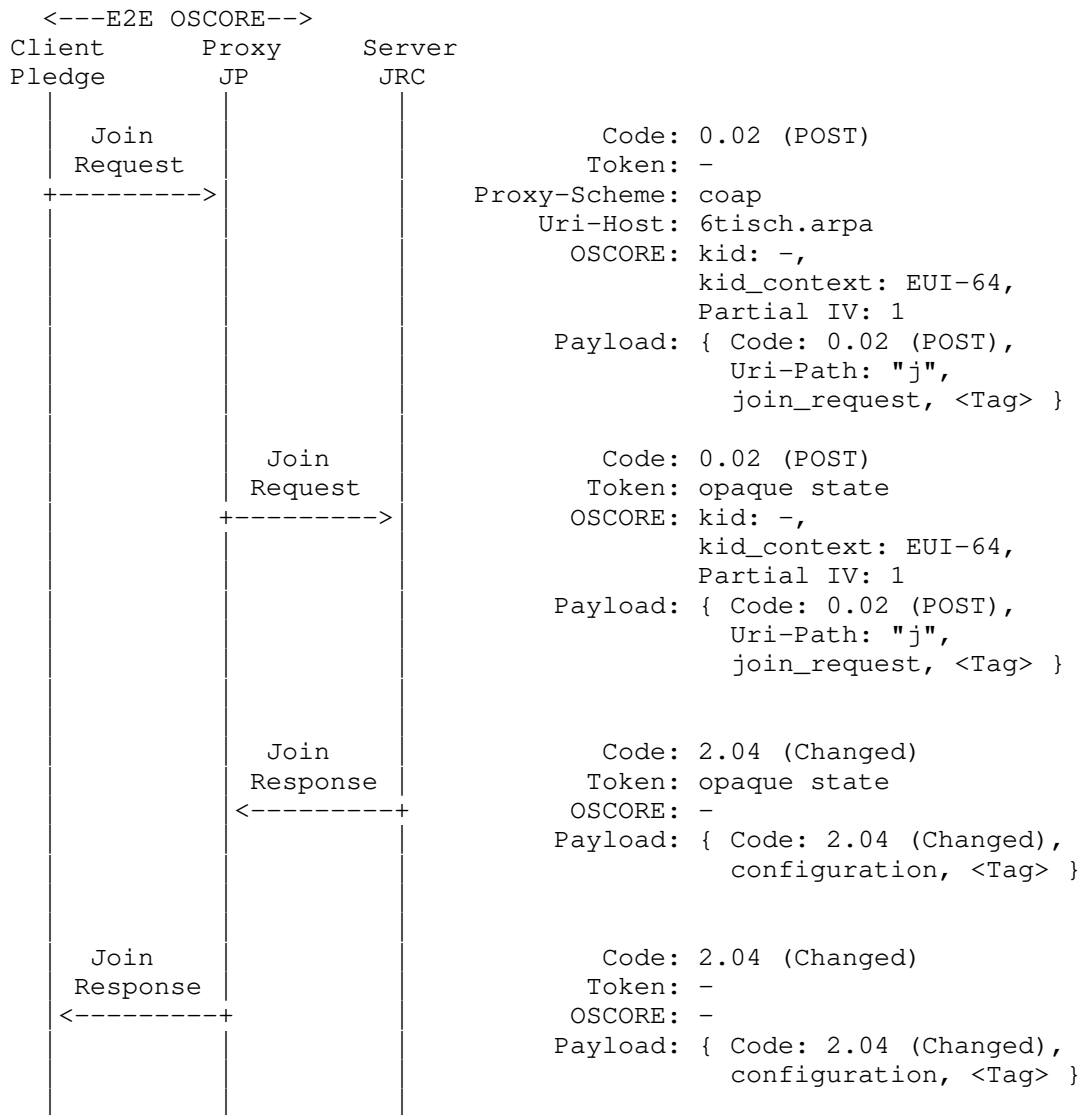


Figure 3: Example of a successful join protocol exchange. { ... } denotes authenticated encryption, <Tag> denotes the authentication tag.

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: Informational
Expires: September 3, 2018

MR. Palattella, Ed.
LIST
P. Thubert
cisco
T. Watteyne
Analog Devices
Q. Wang
Univ. of Sci. and Tech. Beijing
March 2, 2018

Terms Used in IPv6 over the TSCH mode of IEEE 802.15.4e
draft-ietf-6tisch-terminology-10

Abstract

This document provides a glossary of terminology used in IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH). This document extends existing terminology documents for Low-power and Lossy Networks.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 3, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
 2. Terminology 2
 3. Security Considerations 7
 4. References 7
 4.1. Normative References 7
 4.2. Informative References 8
 4.3. External Informative References 8
 Authors' Addresses 8

1. Introduction

The IEEE802.15.4 Medium Access Control (MAC) has evolved with the Time Slotted Channel Hopping (TSCH) mode for industrial-type applications.

This document provides additional terminology elements to cover terms that are new to the context of TSCH wireless networks and other deterministic networks.

2. Terminology

The draft extends [RFC7102] and use terms from [RFC6550] and [RFC6552], which are all included here by reference.

The draft does not reuse terms from IEEE802.15.4 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 802.15.4e): It defines the 6top sublayer, a set of protocols for setting up a TSCH schedule in distributed approach, and a security solution.

6top (6TiSCH Operation Sublayer): The next highest layer of the IEEE802.15.4 TSCH medium access control layer. It implements and terminates 6P, and contains at least one SF.

6P (6top Protocol): Allows neighbor nodes to communicate to add/delete cells to one another in their TSCH schedule.

6P Transaction: Part of 6P, the action of two neighbors exchanging a 6P request message and the corresponding 6P response message.

ASN (Absolute Slot Number): The total number of timeslots that have elapsed since the PAN coordinator has started the TSCH network. Incremented by one at each timeslot. It is wide enough to not roll over in practice.

BBR (Backbone Router): An LBR and also a IPv6 ND-efficiency-aware Router (NEAR) [I-D.chakrabarti-nordmark-6man-efficient-nd]. Performs ND proxy operations between registered devices and classical ND devices that are located on the backbone.

blacklist of frequencies: A set of frequencies which should not be used for communication.

broadcast cell: A scheduled cell used for broadcast transmission.

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.

CCA (Clear Channel Assessment): Mechanism defined in [IEEE802154-2015], section 6.2.5.2. In a TSCH network, CCA can be used to detect other radio networks in vicinity. Nodes listen the channel before sending, to detect other ongoing transmissions. Because the network is synchronized, CCA cannot be used to detect colliding transmission within the same network.

cell: A single element in the TSCH schedule, identified by a slotOffset, a channelOffset, a slotframeHandle. A cell can be scheduled or unscheduled.

centralized cell reservation: A reservation of a cell done by a centralized entity (e.g., a PCE) in the network.

centralized track reservation: A reservation of a track done by a centralized entity (e.g., a PCE) in the network.

Channel Distribution/Usage (CDU) matrix: : 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. It has to be noticed that such a matrix which includes all the cells grouped in chunks, belonging to different slotframes, is different from the TSCH schedule.

channelOffset: Identifies a row in the TSCH schedule. The number of available channelOffset values is equal to the number of available frequencies. The channelOffset translates into a frequency when the communication takes place, resulting in channel hopping.

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).

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 [I-D.ietf-detnet-architecture]. When applied to 6TiSCH, it refers to the reservation of tracks which guarantee an end-to-end latency and optimize the 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 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 cannot relocate.

hopping sequence: Ordered sequence of frequencies, identified by a Hopping_Sequence_ID, used for channel hopping when translating the channel offset 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 [IEEE802154-2015]. Some IE identifiers are managed by the IETF [I-D.kivinen-802-15-ie].

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.

LBR: Low-power Lossy Network (LLN) Border Router. It is an LLN device, usually powered, that acts as a Border Router to the outside within the 6TiSCH architecture.

link: A communication facility or medium over which nodes can communicate at the link layer, the layer immediately below IP. The IETF parlance for the term "Link" is adopted, as opposed to the IEEE802.15.4 terminology.

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, timeskeeping. A scheduled cell can be used by the IEEE802.15.4 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 determined sequence of cells along a multi-hop path. It is typically the result of a track reservation. The node that initializes the process of establishing a track is the owner of the track. The latter assigns a unique identifier to the track, called TrackID.

TrackID: Unique identifier of a track.

TSCH (6top Scheduling Function Identifier): A medium access mode of the [IEEE802154-2015] 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 IEEE802.15.4 TSCH implementation.

3. Security Considerations

Since this document specifies terminology and does not specify new procedures or protocols, it raises no new security issues.

4. References

4.1. Normative References

[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>>.

[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>>.

[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>>.

4.2. Informative References

[I-D.chakrabarti-nordmark-6man-efficient-nd]
Chakrabarti, S., Nordmark, E., Thubert, P., and M. Wasserman, "IPv6 Neighbor Discovery Optimizations for Wired and Wireless Networks", draft-chakrabarti-nordmark-6man-efficient-nd-07 (work in progress), February 2015.

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-04 (work in progress), October 2017.

[I-D.kivinen-802-15-ie]
Kivinen, T. and P. Kinney, "IEEE 802.15.4 Information Element for IETF", draft-kivinen-802-15-ie-06 (work in progress), March 2017.

4.3. External Informative References

[IEEE802154-2015]
IEEE standard for Information Technology, "IEEE Std 802.15.4-2015 Standard for Low-Rate Wireless Personal Area Networks (WPANs)", December 2015.

Authors' Addresses

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

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

Pascal Thubert
Cisco Systems, Inc
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
France

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

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

Email: thomas.watteyne@analog.com

Qin Wang
Univ. of Sci. and Tech. Beijing
30 Xueyuan Road
Beijing 100083
China

Phone: +86 (10) 6233 4781
Email: wangqin@ies.ustb.edu.cn

6lo Working Group
Internet-Draft
Intended status: Informational
Expires: August 12, 2018

D. Dujovne
Universidad Diego Portales
M. Richardson
Sandelman Software Works
February 08, 2018

IEEE802.15.4 Informational Element encapsulation of 6tisch Join and
Enrollment Information
draft-richardson-6tisch-enrollment-enhanced-beacon-00

Abstract

In TSCH mode of IEEE802.15.4, as described by [RFC8180], opportunities for broadcasts are limited to specific times and specific channels. Nodes in a TSCH network typically frequently send Enhanced Beacon (EB) frames to announce the presence of the network. This document provides a mechanism by which small details critical for new nodes (pledges) and long sleeping nodes may be carried within the Enhanced Beacon.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 12, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Terminology	2
1.2. Layer-2 Synchronization	3
1.3. Layer-3 synchronization IPv6 Router solicitations and advertisements	3
2. Protocol Definition	4
2.1. Protocol Example	5
3. Security Considerations	5
4. Privacy Considerations	5
5. IANA Considerations	5
6. Acknowledgements	6
7. References	6
7.1. Normative References	6
7.2. Informative References	7
Appendix A. Change history	7
Authors' Addresses	7

1. Introduction

[RFC7554] describes the use of the time-slotted channel hopping (TSCH) mode of [ieee802154]. As further details in [RFC8180], an Enhanced Beacon is transmitted during a slot designated a broadcast slot.

EDNOTE: Explain why broadcasts are rare, and why we need them. What the Enhanced Beacon is, and what Information Elements are, and how the IETF has a subtype for that area. Explain what kind of things could be placed in Information Elements, how big they could be, and how they could be compressed.

1.1. Terminology

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

1.2. Layer-2 Synchronization

As explained in section 6 of [RFC8180], the Enhanced Beacon has a number of purposes: synchronization of ASN and Join Metric, timeslot template identifier, the channel hopping sequence identifier, TSCH SlotFrame and Link IE.

The Enhanced Beacon (EB) is used by nodes already part of a TSCH network to announce its existence. Receiving an EB allows a Joining Node (pledge) to learn about the network and synchronize to it. The EB may also be used as a means for a node already part of the network to re-synchronize [RFC7554].

There are a limited number of timeslots designated as a broadcast slot by each router. These slots are rare, and with 10ms slots, with a slot-frame length of 100, there may be only 1 slot/s for the beacon.

1.3. Layer-3 synchronization IPv6 Router solicitations and advertisements

At layer 3, [RFC2461] defines a mechanism by which nodes learn about routers by listening for multicasted Router Advertisements (RA). If no RA is heard within a set time, then a Router Solicitation (RS) may be multicast, to which an RA will be received, usually unicast.

Although [RFC6775] reduces the amount of multicast necessary to do address resolution via Neighbor Solicitation messages, it still requires multicast of either RAs or RS. This is an expensive operation for two reasons: there are few multicast timeslots for unsolicited RAs; if a pledge node does not hear an RA, and decides to send a RS (consuming a broadcast aloha slot with unencrypted traffic), many unicast RS may be sent in response.

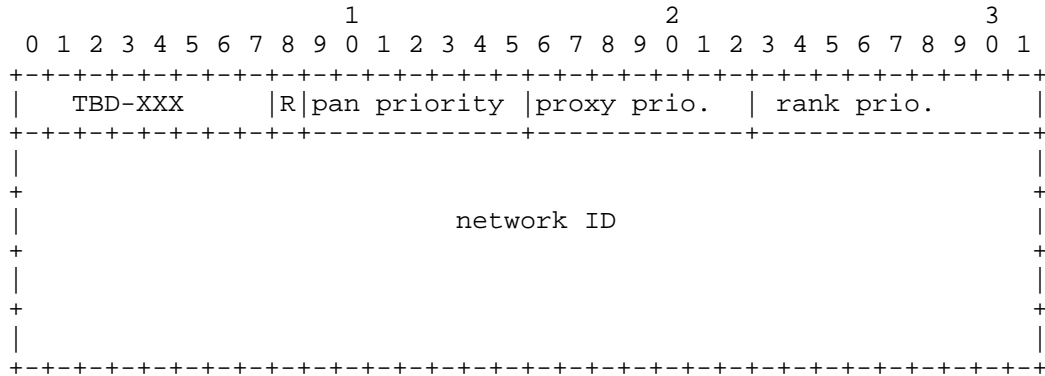
This is a particularly acute issue for the join process for the following reasons:

1. use of a multicast slot by even a non-malicious unauthenticated node for a Router Solicitation may overwhelm that time slot.
2. it may require many seconds of on-time before a new pledge hears a Router Solicitation that it can use.
3. a new pledge may listen to many Enhanced Beacons before it can pick an appropriate network and/or closest Join Assistant to attach to. If it must listen for a RS as well as find the Enhanced Beacon, then the process may take a very long time.

2. Protocol Definition

[RFC8137] creates a registry for new IETF IE subtypes. This document allocates a new subtype TBD-XXX.

This document documents a new IE subtype structure is as follows. As explained in [RFC8137] the length of the Sub-Type Content can be calculated from the container, so no length information is necessary.



proxy priority the proxy priority value contains a number from 0 to 0x7f. Lower numbers are considered to be a higher preference. A priority of 0x7f indicates that the announcer should never be considered as a viable enrollment proxy. Lower value indicates willing to act as a Join Proxy as described in [I-D.ietf-6tisch-minimal-security]. Only unenrolled pledges look at this value.

pan priority the pan priority is a value set by the 6LBR to indicate the relative priority of this LLN compared to those with different PANIDs. This value may be used as part of the enrollment priority, but typically is used by devices which have already enrolled, and need to determine which PAN to pick. Unenrolled pledges MAY consider this value when selecting a PAN to join. Enrolled devices MAY consider this value when looking for an eligible parent device.

rank priority the rank priority is set by the 6LR which sent the beacon and is an indication of how willing this 6LR is to serve as an RPL parent within a particular network ID. This is a local value to be determined in other work. It might be calculated from RPL rank, and it may include some modifications based upon current number of children, or number of neighbor cache entries available. This value MUST be ignored by pledges, it is for enrolled devices only.

R the Router Advertisement flag is set if the sending node will act as a Router for host-only nodes that need addressing via unicast Router Solicitation messages.

network ID this is an opaque 16-byte identifier that uniquely identifies this network, potentially among many networks that are operating in the same frequencies in overlapping physical space.

In a 6tisch network, where RPL is used as the mesh routing protocol, the network ID can be constructed from a SHA256 hash of the prefix (/64) of the network. That is just a suggestion for a default value. In some LLNs where multiple PANIDs may lead to the same management device (the JRC), then a common value that is the same across all PANs MUST be configured.

2.1. Protocol Example

Here will be three examples of processing.

3. Security Considerations

All of the contents of this Information Element are sent in the clear. The containing Enhanced Beacon is not encrypted.

The Enhanced Beagon is authenticated at the layer-2 level using 802.15.4 mechanisms using the network-wide keying material. Nodes which are enrolled will have the network-wide keying material and can validate the beacon.

Pledges which have not yet enrolled are unable to authenticate the beacons.

4. Privacy Considerations

The use of a network ID may reveal information about the network. The use of a SHA256 hash of the DODAGID, rather than using the DODAGID directly provides some cover the addresses used within the network. The DODAGID is usually the IPv6 address of the root of the RPL mesh.

An interloper with a radio sniffer would be able to use the network ID to map out the extend of the mesh network.

5. IANA Considerations

Allocate a new number TBD-XXX from Registry IETF IE Sub-type ID. This entry should be called 6tisch-Join-Info.

6. Acknowledgements

Thomas Watteyne provided extensive editorial comments on the document.

7. References

7.1. 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-13 (work in progress), November 2017.

[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-04 (work in progress), October 2017.

[ieee802154]

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

[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>>.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, DOI 10.17487/RFC2461, December 1998, <<https://www.rfc-editor.org/info/rfc2461>>.

[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>>.

[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>>.

[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

[I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.

[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. Change history

The extension was originally for the use of Pledges only during the enrollment/join process. Additional information was desired for nodes which have already enrolled in order to aid in the joining (selecting of a parent) of an RPL DAG. The term "join" was realized to be ambiguous, meaning different things to different groups, and so the activity where the pledge finds a "Join Proxy" has been named "enrollment"

This is an evolution of an earlier proposal which provided for storing an entire IPv6 Router Advertisement in an Informational Element. It was deemed too general a solution, possibly subject to mis-use. This proposal restricts the use to just the key pieces of information required.

Authors' Addresses

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

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

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

6lo Working Group
Internet-Draft
Intended status: Informational
Expires: October 26, 2018

D. Dujovne
Universidad Diego Portales
M. Richardson
Sandelman Software Works
April 24, 2018

IEEE802.15.4 Informational Element encapsulation of 6tisch Join and
Enrollment Information
draft-richardson-6tisch-enrollment-enhanced-beacon-01

Abstract

In TSCH mode of IEEE802.15.4, as described by [RFC8180], opportunities for broadcasts are limited to specific times and specific channels. Nodes in a TSCH network typically frequently send Enhanced Beacon (EB) frames to announce the presence of the network. This document provides a mechanism by which small details critical for new nodes (pledges) and long sleeping nodes may be carried within the Enhanced Beacon.

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 October 26, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Terminology	2
1.2. Layer-2 Synchronization	3
1.3. Layer-3 synchronization IPv6 Router solicitations and advertisements	3
2. Protocol Definition	4
2.1. Protocol Example	5
3. Security Considerations	5
4. Privacy Considerations	6
5. IANA Considerations	6
6. Acknowledgements	6
7. References	6
7.1. Normative References	6
7.2. Informative References	7
Appendix A. Change history	7
Authors' Addresses	8

1. Introduction

[RFC7554] describes the use of the time-slotted channel hopping (TSCH) mode of [ieee802154]. As further details in [RFC8180], an Enhanced Beacon is transmitted during a slot designated a broadcast slot.

EDNOTE: Explain why broadcasts are rare, and why we need them. What the Enhanced Beacon is, and what Information Elements are, and how the IETF has a subtype for that area. Explain what kind of things could be placed in Information Elements, how big they could be, and how they could be compressed.

1.1. Terminology

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

1.2. Layer-2 Synchronization

As explained in section 6 of [RFC8180], the Enhanced Beacon has a number of purposes: synchronization of ASN and Join Metric, timeslot template identifier, the channel hopping sequence identifier, TSCH SlotFrame and Link IE.

The Enhanced Beacon (EB) is used by nodes already part of a TSCH network to announce its existence. Receiving an EB allows a Joining Node (pledge) to learn about the network and synchronize to it. The EB may also be used as a means for a node already part of the network to re-synchronize [RFC7554].

There are a limited number of timeslots designated as a broadcast slot by each router. These slots are rare, and with 10ms slots, with a slot-frame length of 100, there may be only 1 slot/s for the beacon.

1.3. Layer-3 synchronization IPv6 Router solicitations and advertisements

At layer 3, [RFC2461] defines a mechanism by which nodes learn about routers by listening for multicasted Router Advertisements (RA). If no RA is heard within a set time, then a Router Solicitation (RS) may be multicast, to which an RA will be received, usually unicast.

Although [RFC6775] reduces the amount of multicast necessary to do address resolution via Neighbor Solicitation messages, it still requires multicast of either RAs or RS. This is an expensive operation for two reasons: there are few multicast timeslots for unsolicited RAs; if a pledge node does not hear an RA, and decides to send a RS (consuming a broadcast aloha slot with unencrypted traffic), many unicast RS may be sent in response.

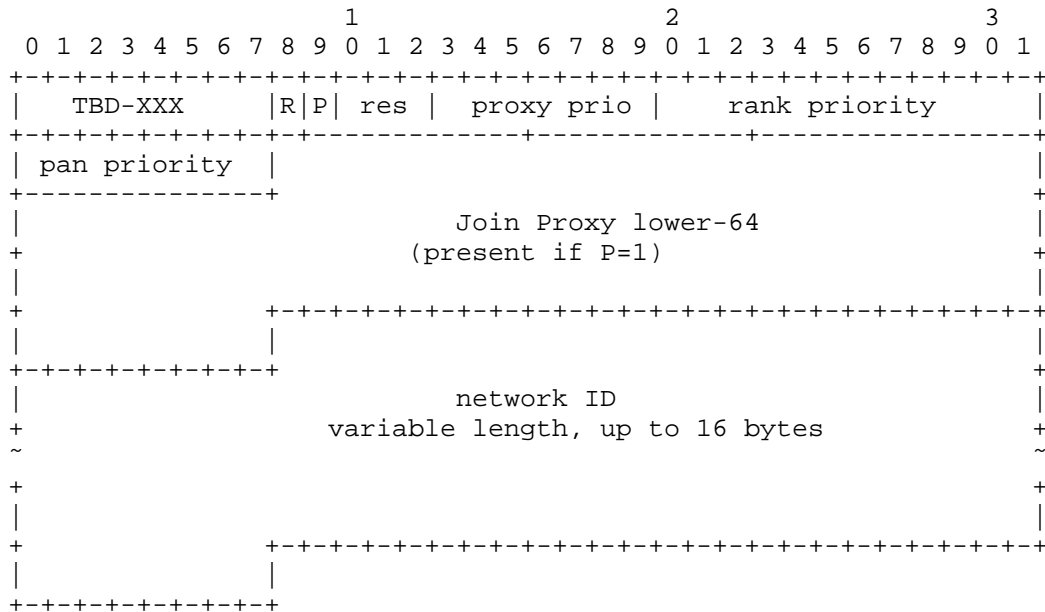
This is a particularly acute issue for the join process for the following reasons:

1. use of a multicast slot by even a non-malicious unauthenticated node for a Router Solicitation may overwhelm that time slot.
2. it may require many seconds of on-time before a new pledge hears a Router Solicitation that it can use.
3. a new pledge may listen to many Enhanced Beacons before it can pick an appropriate network and/or closest Join Assistant to attach to. If it must listen for a RS as well as find the Enhanced Beacon, then the process may take a very long time.

2. Protocol Definition

[RFC8137] creates a registry for new IETF IE subtypes. This document allocates a new subtype TBD-XXX.

This document documents a new IE subtype structure is as follows. As explained in [RFC8137] the length of the Sub-Type Content can be calculated from the container, so no length information is necessary.



proxy priority the proxy prority value contains a number from 0 to 0x7f. Lower numbers are considered to be a higher preference. A priority of 0x7f indicates that the announcer should never be considered as a viable enrollment proxy. Lower value indicates willing to act as a Join Proxy as described in [I-D.ietf-6tisch-minimal-security]. Only unenrolled pledges look at this value.

pan priority the pan priority is a value set by the DODAG root to indicate the relative priority of this LLN compared to those with different PANIDs. This value may be used as part of the enrollment priority, but typically is used by devices which have already enrolled, and need to determine which PAN to pick. Unenrolled pledges MAY consider this value when selecting a PAN to join. Enrolled devices MAY consider this value when looking for an eligible parent device.

rank priority the rank "priority" is set by the 6LR which sent the beacon and is an indication of how willing this 6LR is to serve as an RPL parent within a particular network ID. This is a local value to be determined in other work. It might be calculated from RPL rank, and it may include some modifications based upon current number of children, or number of neighbor cache entries available. This value MUST be ignored by pledges, it is for enrolled devices only.

R the Router Advertisement flag is set if the sending node will act as a Router for host-only nodes that need addressing via unicast Router Solicitation messages.

P if the Proxy Address bit is set, then the lower 64-bits of the Join Proxy's Link Layer address follows the network ID. If the Proxy Address bit is not set, then the Link Layer address of the Join Proxy is identical to the Layer-2 8-byte address used to originate this enhanced beacon. In either case, the layer-2 address of any IPv6 traffic to the originator of this beacon may use the layer-2 address which was used to originate the beacon.

join-proxy lower-64 if the P bit is set, then 64 bits (8 bytes) of address are present. The Link Layer address of the Join Proxy is fe80 (as for any Link Layer address), and the bits given in this field.

network ID this is a variable length field, up to 16-bytes in size that uniquely identifies this network, potentially among many networks that are operating in the same frequencies in overlapping physical space. The length of this field can be calculated as being whatever is left in the Information Element.

In a 6tisch network, where RPL is used as the mesh routing protocol, the network ID can be constructed from a SHA256 hash of the prefix (/64) of the network. That is just a suggestion for a default value. In some LLNs where multiple PANIDs may lead to the same management device (the JRC), then a common value that is the same across all PANs MUST be configured.

2.1. Protocol Example

Here will be three examples of processing.

3. Security Considerations

All of the contents of this Information Element are sent in the clear. The containing Enhanced Beacon is not encrypted.

The Enhanced Beacon is authenticated at the layer-2 level using 802.15.4 mechanisms using the network-wide keying material. Nodes which are enrolled will have the network-wide keying material and can validate the beacon.

Pledges which have not yet enrolled are unable to authenticate the beacons.

4. Privacy Considerations

The use of a network ID may reveal information about the network. The use of a SHA256 hash of the DODAGID, rather than using the DODAGID directly provides some cover the addresses used within the network. The DODAGID is usually the IPv6 address of the root of the RPL mesh.

An interloper with a radio sniffer would be able to use the network ID to map out the extend of the mesh network.

5. IANA Considerations

Allocate a new number TBD-XXX from Registry IETF IE Sub-type ID. This entry should be called 6tisch-Join-Info.

6. Acknowledgements

Thomas Watteyne provided extensive editorial comments on the document.

7. References

7.1. 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-13 (work in progress), November 2017.

[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-05 (work in progress), March 2018.

[ieee802154]

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

- [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>>.
- [RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, DOI 10.17487/RFC2461, December 1998, <<https://www.rfc-editor.org/info/rfc2461>>.
- [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>>.
- [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>>.
- [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

- [I-D.ietf-6tisch-dtsecurity-secure-join] Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.
- [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. Change history

The rank priority was expanded to 2 bytes.

00: The extension was originally for the use of Pledges only during the enrollment/join process. Additional information was desired for nodes which have already enrolled in order to aid in the joining (selecting of a parent) of an RPL DAG. The term "join" was realized to be ambiguous, meaning different things to different groups, and so

the activity where the pledge finds a "Join Proxy" has been named "enrollment"

-1: This is an evolution of an earlier proposal which provided for storing an entire IPv6 Router Advertisement in an Informational Element. It was deemed too general a solution, possibly subject to mis-use. This proposal restricts the use to just the key pieces of information required.

Authors' Addresses

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

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

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

6lo Working Group
Internet-Draft
Intended status: Informational
Expires: August 6, 2018

M. Richardson
Sandelman Software Works
February 02, 2018

Enabling secure network enrollment in RPL networks
draft-richardson-6tisch-roll-enrollment-priority-00

Abstract

[I-D.richardson-6tisch-join-enhanced-beacon] defines a method by which a potential [I-D.ietf-6tisch-minimal-security] can announce itself as a available for new Pledges to Join a network. The announcement includes a priority for join. This document provides a mechanism by which a RPL DODAG root can disable join announcements, or adjust the base priority for join operation.

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 August 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Protocol Definition	3
3. Security Considerations	4
4. Privacy Considerations	4
5. IANA Considerations	4
6. Acknowledgements	4
7. References	4
7.1. Normative References	4
7.2. Informative References	5
Appendix A. Change history	6
Author's Address	6

1. Introduction

[RFC7554] describes the use of the time-slotted channel hopping (TSCH) mode of [ieee802154]. [I-D.ietf-6tisch-minimal-security] and [I-D.ietf-6tisch-dtsecurity-secure-join] describe mechanisms by which a new node (the "pledge") can use a friendly router as a Join Proxy. [I-D.richardson-6tisch-join-enhanced-beacon] describes an extension to the 802.15.4 Enhanced Beacon that is used by a Join Proxy to announce its existence such that Pledges can find them.

It has become clear that not every routing member of the mesh ought to announce itself as a Join Proxy. There are a variety of local reasons by which a 6LR might not want to provide the Join Proxy function. They include available battery power, already committed network bandwidth, and also total available memory available for Join proxy neighbor cache slots.

There are other situations where the operator of the network would like to selective enable or disable the join process in a particular DODAG.

As the join process involves permitting unencrypted traffic into the best effort part of a (TSCH) network, it would be better to have the join process off when no new nodes are expected.

A network operator might also be able to recognize when certain parts of the network are overloaded and can not accomodate additional join traffic, and it would like to adjust the join priority among all nodes in the subtree of a congested link.

This document describes an RPL DIO option that can be used to announce a minimum join priority.

1.1. Terminology

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

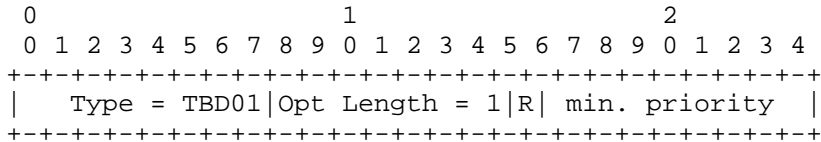
In addition, the terminology of [I-D.ietf-6tisch-terminology] and from [I-D.ietf-anima-voucher] are used.

2. Protocol Definition

The following option is defined to transmission in the DIO issued by the DODAG root. It may also be added by a router on part of the subtree as a result of some (out of scope for this document) management function.

6LRs that see this DIO Option SHOULD increment the minimum priority if they observe congestion on the channel used for join traffic. (TODO: how much? Do we need to standardize this?)

A 6LR which would otherwise be willing to act as a Join Proxy, will examine the minimum priority field, and to that number, add any additional local consideration (such as upstream congestion). The resulting priority, if less than 0x7f should enable the Join Proxy function.



min.priority a 7 bit field which provides a base value for the Enhanced Beacon Join priority. A value of 0x7f (127) disables the Join Proxy function entirely.

R a reserved bit that SHOULD be set to 0 by senders, and MUST be ignored by receivers. The reserved bit SHOULD be copied to options created.

3. Security Considerations

As per [RFC7416], RPL control frames either run over a secured layer 2, or use the [RFC6550] Secure DIO methods. This option can be placed into either a "clear" (layer-2 secured) DIO, or a layer-3 Secure DIO. As such this option will have both integrity and confidentiality mechanisms applied to it.

A malicious node (that was part of the RPL control plane) could see these options and could, based upon the observed minimal join priority signal a confederate that it was a good time to send malicious join traffic.

A malicious node (that was part of the RPL control plane) could also send DIOs with a different minimal join priority which would cause downstream mesh routers to change their Join Proxy behaviour. Lower minimal priorities would cause downstream nodes to accept more pledges than the network was expecting, and higher minimal priorities cause the join process to stall.

The use of layer-2 or layer-3 security for RPL control messages prevents the above two attacks.

4. Privacy Considerations

There are no new privacy issues caused by this extension.

5. IANA Considerations

Allocate a new number TBD01 from Registry RPL Control Message Options. This entry should be called Minimum Join Priority.

6. Acknowledgements

none so far.

7. References

7.1. 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-04 (work in progress), October
2017.

- [I-D.richardson-6tisch-join-enhanced-beacon]
Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational Element encapsulation of 6tisch Join Information", draft-richardson-6tisch-join-enhanced-beacon-03 (work in progress), January 2018.
- [ieee802154]
IEEE Standard, ., "802.15.4-2015 - IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015, <<http://standards.ieee.org/findstds/standard/802.15.4-2015.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., and M. Richardson, Ed., "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015, <<https://www.rfc-editor.org/info/rfc7416>>.
- [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>>.

7.2. Informative 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-13 (work in progress), November 2017.
- [I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.

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

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

[I-D.ietf-anima-voucher]

Watson, K., Richardson, M., Pritikin, M., and T. Eckert,
"Voucher Profile for Bootstrapping Protocols", draft-ietf-
anima-voucher-07 (work in progress), January 2018.

[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>>.

Appendix A. Change history

version 00.

Author's Address

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

6lo Working Group
Internet-Draft
Intended status: Informational
Expires: February 9, 2020

M. Richardson
Sandelman Software Works
August 08, 2019

Enabling secure network enrollment in RPL networks
draft-richardson-6tisch-roll-enrollment-priority-03

Abstract

[I-D.ietf-6tisch-enrollment-enhanced-beacon] defines a method by which a potential [I-D.ietf-6tisch-minimal-security] join proxy can announce itself as a available for new Pledges to Join a network. The announcement includes a priority for join. This document provides a mechanism by which a RPL DODAG root can disable join announcements, or adjust the base priority for join operation.

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 February 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Protocol Definition	3
3. Security Considerations	4
4. Privacy Considerations	4
5. IANA Considerations	4
6. Acknowledgements	4
7. References	4
7.1. Normative References	4
7.2. Informative References	5
Appendix A. Change history	6
Author's Address	6

1. Introduction

[RFC7554] describes the use of the time-slotted channel hopping (TSCH) mode of [ieee802154]. [I-D.ietf-6tisch-minimal-security] and [I-D.ietf-6tisch-dtsecurity-secure-join] describe mechanisms by which a new node (the "pledge") can use a friendly router as a Join Proxy. [I-D.ietf-6tisch-enrollment-enhanced-beacon] describes an extension to the 802.15.4 Enhanced Beacon that is used by a Join Proxy to announce its existence such that Pledges can find them.

It has become clear that not every routing member of the mesh ought to announce itself as a Join Proxy. There are a variety of local reasons by which a 6LR might not want to provide the Join Proxy function. They include available battery power, already committed network bandwidth, and also total available memory available for Join proxy neighbor cache slots.

There are other situations where the operator of the network would like to selective enable or disable the join process in a particular DODAG.

As the join process involves permitting unencrypted traffic into the best effort part of a (TSCH) network, it would be better to have the join process off when no new nodes are expected.

A network operator might also be able to recognize when certain parts of the network are overloaded and can not accomodate additional join traffic, and it would like to adjust the join priority among all nodes in the subtree of a congested link.

This document describes an RPL DIO option that can be used to announce a minimum join priority. Each potential Join Proxy would this value as a base on which to add (decreasing likely hood of attracting traffic) values relating to local conditions.

A network operator can set this value to the maximum value allowed, effectively disable all new join traffic.

1.1. Terminology

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

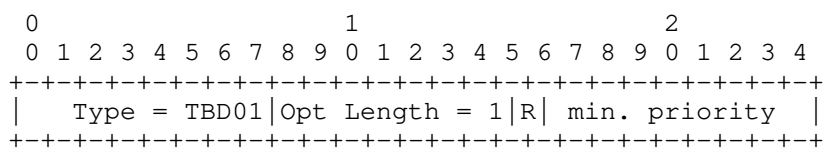
In addition, the terminology of [I-D.ietf-6tisch-terminology] and from [RFC8366] are used.

2. Protocol Definition

The following option is defined to transmission in the DIO issued by the DODAG root. It may also be added by a router on part of the sub-tree as a result of some (out of scope for this document) management function.

6LRs that see this DIO Option SHOULD increment the minimum priority if they observe congestion on the channel used for join traffic. (TODO: how much? Do we need to standardize this?)

A 6LR which would otherwise be willing to act as a Join Proxy, will examine the minimum priority field, and to that number, add any additional local consideration (such as upstream congestion). The resulting priority, if less than 0x7f should enable the Join Proxy function.



min.priority a 7 bit field which provides a base value for the Enhanced Beacon Join priority. A value of 0x7f (127) disables the Join Proxy function entirely.

R a reserved bit that SHOULD be set to 0 by senders, and MUST be ignored by receivers. The reserved bit SHOULD be copied to options created.

3. Security Considerations

As per [RFC7416], RPL control frames either run over a secured layer 2, or use the [RFC6550] Secure DIO methods. This option can be placed into either a "clear" (layer-2 secured) DIO, or a layer-3 Secure DIO. As such this option will have both integrity and confidentiality mechanisms applied to it.

A malicious node (that was part of the RPL control plane) could see these options and could, based upon the observed minimal join priority signal a confederate that it was a good time to send malicious join traffic.

A malicious node (that was part of the RPL control plane) could also send DIOs with a different minimal join priority which would cause downstream mesh routers to change their Join Proxy behaviour. Lower minimal priorities would cause downstream nodes to accept more pledges than the network was expecting, and higher minimal priorities cause the join process to stall.

The use of layer-2 or layer-3 security for RPL control messages prevents the above two attacks.

4. Privacy Considerations

There are no new privacy issues caused by this extension.

5. IANA Considerations

Allocate a new number TBD01 from Registry RPL Control Message Options. This entry should be called Minimum Join Priority.

6. Acknowledgements

This has been reviewed by Pascal Thubert and Thomas Wattenye.

7. References

7.1. Normative References

- [I-D.ietf-6tisch-enrollment-enhanced-beacon]
Dujovne, D. and M. Richardson, "IEEE802.15.4 Informational Element encapsulation of 6tisch Join and Enrollment Information", draft-ietf-6tisch-enrollment-enhanced-beacon-02 (work in progress), March 2019.
- [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-12 (work in progress), July 2019.
- [ieee802154]
IEEE Standard, ., "802.15.4-2015 - IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", 2015, <<http://standards.ieee.org/findstds/standard/802.15.4-2015.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC7416] Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., and M. Richardson, Ed., "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)", RFC 7416, DOI 10.17487/RFC7416, January 2015, <<https://www.rfc-editor.org/info/rfc7416>>.
- [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>>.

7.2. Informative 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-24 (work in progress), July 2019.

- [I-D.ietf-6tisch-dtsecurity-secure-join]
Richardson, M., "6tisch Secure Join protocol", draft-ietf-6tisch-dtsecurity-secure-join-01 (work in progress), February 2017.
- [I-D.ietf-6tisch-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.
- [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>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.

Appendix A. Change history

version 00.

Author's Address

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

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

X. Vilajosana, Ed.
P. Tuset
B. Martinez
Universitat Oberta de Catalunya
J. Munoz
Inria
March 1, 2018

Global Time Distribution in 6TiSCH Networks
draft-vilajosana-6tisch-globaltime-00

Abstract

This specification defines an optional extension to the Join Response message defined by the Minimal Security Framework for 6TiSCH. The extension aims at providing global time distribution support so nodes in the 6TiSCH network can exploit global time information instead of relying only in relative network time based on the Absolute Sequence Number (ASN). The specification also defines a mechanism for resynchronization, to handle leap seconds or to enable periodic global time updates relying on a CoAP service.

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 [RFC2119].

Status of This Memo

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

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

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

This Internet-Draft will expire on September 2, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Global Time Source	3
3. Global Time Extension to the Join Response	4
4. Resynchronization	6
5. Leap Second handling	6
6. References	7
6.1. Normative References	7
6.2. Informative References	7
Authors' Addresses	8

1. Introduction

Time Synchronized Channel Hopping (TSCH) exploits node synchronization to build up deterministic access networks through scheduling [RFC7554]. 6TiSCH defines a control plane architecture to enable IEEE802.15.4 TSCH networks to securely bootstrap and in a distributed manner self-organize in order to meet application traffic needs [I-D.ietf-6tisch-architecture]. The synchronization accuracy between the nodes' clocks in a 6TiSCH network is dependent on the network maintenance traffic (Keep Alives), application traffic, and MAC layer guard time duration. It is well-known that the smaller the guard time, the smaller the tolerated drift between two nodes, and hence the more precise their synchronization.

The concept of network synchronization is achieved through a virtual counter referred as Absolute Sequence Number (ASN). In a 6TiSCH network, each node updates its ASN at every slot, giving the nodes the same notion of time (with timeslot granularity). This time is relative to the moment the network started or reset and hence cannot be used to compare tagged events from different networks.

This document defines a data structure to map ASN and absolute time. The document then defines the procedure to transport the structure to the 6TiSCH nodes:

- o As an optional extension to the Join Response procedure within the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security].
- o As a CoAP Response [RFC7252] to a global time service exposed by a node in the network and announced through the Join Response.

2. Global Time Source

In order to distribute global time information in a 6TiSCH network at least one component must be acting as a global time source and enabling nodes in the network to obtain the absolute time reference from it. The way global time is obtained and maintained in this network component is out of scope of this specification. As an example, this component can account for the global time in the network internally, can use an external source to obtain global time (e.g., GPS, NTP [RFC5905]) or, can be synchronized through a precision time protocol such the IEEE-1588 [IEEE1588] to another network.

We use the example network in Figure 1 throughout this specification for illustration. The Join Registrar/Coordinator (JRC) acts as the global time information source for a node when it joins. How the JRC obtains such global time information is out of scope of this specification. This specification defines how the JRC formats and distributes the absolute time reference to the 6TiSCH nodes in the network.

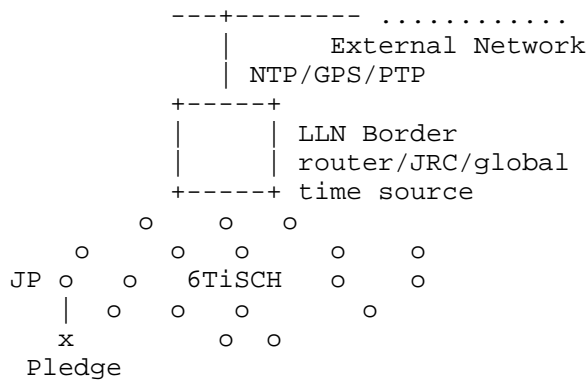


Figure 1: An example network

A Pledge node obtains its global time reference during the Secure Join Process with the JRC [I-D.ietf-6tisch-minimal-security]. This specification extends the Join Response message with an optional data structure which includes the global time reference and an optional CoAP link-format and IPv6 address pointing to the network global time service to support periodic absolute time updates.

The global time reference is a mapping between the ASN of the network and the global time at the moment of processing the Join Response. After having obtained the global time reference, a 6TiSCH node maintains internally its timing until it resets or is disconnected from the network. Optionally, periodic refresh messages can be issued by the 6TiSCH node to the node that acts as global time source, through a provided CoAP URI exposing the time service. The JRC or any other node in the network can expose that service.

3. Global Time Extension to the Join Response

This document extended the Join Response defined in [I-D.ietf-6tisch-minimal-security] with:

- o A byte string containing the ASN at which the CoAP Response (e.g, Join Response) is processed at the JRC. The 5-byte ASN is carried in network byte order.
- o An 8-bit unsigned integer containing an era counter.
- o A 32-bit unsigned integer of containing a timestamp in seconds, captured at the beginning of the timeslot at which the CoAP Response (e.g. Join Response) is processed. Carried in network byte order.
- o A 32-bit unsigned integer containing the number of picoseconds elapsed after the last entire second at the beginning of the timeslot at which the CoAP Response (e.g. Join Response) is processed. Carried in network byte order.
- o Optionally, a byte string encoding the IPv6 address of the global time source.
- o Optionally, a byte string encoding a global time service URI in core-link format.
- o Optionally, an unsigned word lease value indicating the number of days of freshness of the assigned global time information.

```

global_time_option = [
    asn: bstr,
    era: uint8,
    seconds: uint32,
    fraction: uint32,
    ? gt_address : bstr,
    ? gt_service: link-format (bstr),
    ? gt_lease: uint16,
]

```

To take into account possible leap seconds. A `leap_second_option` is defined by:

- o An 8-bit unsigned integer containing the action to be performed when the next leap second day is reached.
- o A 16-bit unsigned integer containing an offset in days to the beginning of the day (0 h UTC) when the next leap second must be applied. Carried in network byte order.

```

leap_second_option = [
    leap_indicator: uint8,
    leap_offset: uint16,
]

```

The era counter is used to account for wraps of the seconds counter. It starts at 0 and increments approximately every 136 years as per definition of era in [RFC5905]. Era 0 starts at 0h UTC 1st of January 1900. The seconds and fraction fields are based on the specification described in the [RFC5905] for the Timestamp format. The seconds field accounts for seconds elapsed since the 0h on the 1 January 1900 UTC, as described by the [RFC5905]. The fraction field provides synchronization precisions in the order of hundreds of picoseconds. Its granularity is described in [RFC5905]. If the global time service is co-located at the JRC, the `gt_address` field can be omitted as the address is known through the Join Response. An optional byte string, indicating the global time synchronization service URI MAY be present. The URI is defined using the CORE link-format as per [RFC6690]. An optional `gt_lease` value in days indicating the mandatory refresh period MAY be present. If this value is 0, a node does not refresh the global time information.

The optional `leap_second_option` defines a leap second indicator, which identifies the type of correction that needs to be applied once the next leap second day is reached. The types are described in Figure 9 of [RFC5905]. A `leap_offset` contains the offset in days to when the next leap second needs to be applied, following the action described in the leap second indicator.

The `global_time_option` and the `leap_second_option`, if present, SHOULD be appended following the Join Response Payload and MUST be encoded as a CBOR array object [RFC7049]. A CBOR array contains as a first element the number of items contained by the array. This element then enables to determine if the optional elements are present in the option.

4. Resynchronization

When a pledge receives the Join Response containing the `global_time_option`, it updates its internal absolute time clock/counter. If present, it also stores the `gt_address`, the link-format URI and the lease time.

After correcting a leap second or when the lease period is reached, a node MAY want to update the global time information values to keep track of the next leap second correction event or to renew its global time synchronization lease. This resynchronization is conducted through a CoAP GET Request to the `gt_address` and `gt_service` URI.

- o The request method is GET.
- o The type is Non-confirmable (NON).
- o The Proxy-Scheme option is set to "coap".
- o The Uri-Host option is defined by the `gt_address`.
- o The Uri-Path option is set to `gt_service`.
- o The payload is empty.

The response is a CoAP Response Message with Response Code 2.05 (Content) containing the `global_time_option` as payload. The response MAY contain a `leap_second_option` in case a leap second update is needed. Both options if present are encoded as CBOR arrays.

5. Leap Second handling

When a 6TiSCH node receives a global time synchronization message, either being a Join Response or a CoAP Response message for a time synchronization Request, and this response contains a `leap_second_option`, the node MUST store the values until the leap second offset is reached.

When a leap second offset is reached, the leap second is corrected adding or subtracting a second to the last minute of the day as indicated by the `leap_indicator` field.

6. References

6.1. 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-04 (work in progress), October
2017.
- [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>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch,
"Network Time Protocol Version 4: Protocol and Algorithms
Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010,
<<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
<<https://www.rfc-editor.org/info/rfc6690>>.
- [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>>.
- [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>>.

6.2. Informative 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-13 (work
in progress), November 2017.

[IEEE1588]

IEEE standard for Information Technology, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) , vol., no., pp.1-300", July 2008.

Authors' Addresses

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

Email: xvilajosana@uoc.edu

Pere Tuset
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Email: peretuset@uoc.edu

Borja Martinez
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Email: bmartinezh@uoc.edu

Jonathan Munoz
Inria
2 rue Simone Iff
Paris 75012
France

Email: jonathan.munoz@inria.fr

6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: December 21, 2018

X. Vilajosana, Ed.
P. Tuset
B. Martinez
Universitat Oberta de Catalunya
J. Munoz
Inria
June 19, 2018

Global Time Distribution in 6TiSCH Networks
draft-vilajosana-6tisch-globaltime-01

Abstract

This specification defines an optional extension to the Constrained Join Protocol (CoJP) defined by the Minimal Security Framework for 6TiSCH. The extension aims at providing global time distribution support so nodes in the 6TiSCH network can exploit global time information instead of relying only in relative network time based on the Absolute Sequence Number (ASN). The specification also defines a mechanism for resynchronization, to handle leap seconds or to enable periodic global time updates relying on a CoAP service.

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 [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 21, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Global Time Source	3
3. Global Time Extension to the Join Response	4
4. Resynchronization	7
5. Leap Second handling	8
6. Security Considerations	8
7. References	8
7.1. Normative References	8
7.2. Informative References	9
Authors' Addresses	9

1. Introduction

Time Synchronized Channel Hopping (TSCH) exploits node synchronization to build up deterministic access networks through scheduling [RFC7554]. 6TiSCH defines a control plane architecture to enable IEEE802.15.4 TSCH networks to securely bootstrap and in a distributed manner self-organize in order to meet application traffic needs [I-D.ietf-6tisch-architecture]. The synchronization accuracy between the nodes' clocks in a 6TiSCH network is dependent on the network maintenance traffic (Keep Alives), application traffic, and MAC layer guard time duration. It is well-known that, for a given traffic, the smaller the guard time, the smaller the tolerated drift between two nodes, and hence, the more precise their synchronization.

The concept of network synchronization is achieved through a virtual counter referred as Absolute Sequence Number (ASN). In a 6TiSCH network, each node updates its ASN at every slot, giving the nodes the same notion of time (with timeslot granularity). This time is relative to the moment the network started or reset and hence cannot be used to compare tagged events from different networks.

This document defines a data structure to map ASN and absolute time. The document then defines the procedure to transport the structure to the 6TiSCH nodes:

- o As an optional extension to the Constrained Join Protocol (CoJP) Join Response procedure within the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security].
- o As a CoAP Response [RFC7252] to a global time service exposed by the Join Registrar/Cordinator (JRC).

2. Global Time Source

In order to distribute global time information in a 6TiSCH network at least one component must be acting as a global time source and enabling nodes in the network to obtain the absolute time reference from it. The way global time is obtained and maintained in this network component is out of scope of this specification. As an example, this component can account for the global time in the network internally, can use an external source to obtain global time (e.g. GPS, NTP [RFC5905]) or, can be synchronized through a precision time protocol (PTP) such the IEEE-1588 [IEEE1588] to another network.

We use the example network in Figure 1 throughout this specification for illustration. The Join Registrar/Coordinator (JRC) acts as the global time information source for a node when it joins. How the JRC obtains such global time information is out of scope of this specification. The JRC can be a component outside of the 6TiSCH network. However it is required to be synchronize to it through the ASN. How the JRC obtains such synchronization to the 6TiSCH network is out of the scope of this specification. This specification defines how the JRC formats and distributes the absolute time reference to the 6TiSCH nodes in the network.

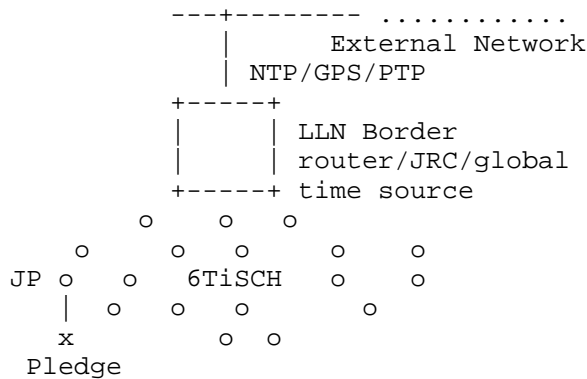


Figure 1: An example network

A Pledge node obtains its global time reference during the Secure Join Process with the JRC [I-D.ietf-6tisch-minimal-security]. This specification extends the Join Response message with an optional data structure which includes the global time reference and optionally the period for absolute time updates.

The global time reference is a mapping between the ASN of the network and the global time at the moment of processing the Join Response. After having obtained the global time reference, a 6TiSCH node maintains internally its timing until it updates it, resets or is disconnected from the network. Optionally, periodic refresh messages can be issued by the 6TiSCH node to the JRC using the JRC URI exposing the time service. These optional refresh messages MAY be used to cope with the clock drift caused by any possible imprecision between the configured timeslot length and the actual timeslot length. As an example, a timeslot can be configured to be 10ms long but because of the nature of the crystal in the device the timeslot becomes 10.1ms.

3. Global Time Extension to the Join Response

This document extends the Join Response message within the Constrained Join Protocol (CoJP) defined in [I-D.ietf-6tisch-minimal-security] with:

- o A byte string containing the ASN at which the CoAP Response (e.g, Join Response) is processed at the JRC. The 5-byte ASN is carried in network byte order.
- o An 8-bit unsigned integer containing an era counter. The era counter is used to account for wraps of the seconds counter. It starts at 0 and increments approximately every 136 years as per

definition of era in NTP [RFC5905]. Era 0 starts at 0h UTC 1st of January 1900.

- o A 32-bit unsigned integer containing a timestamp in seconds, captured at the beginning of the timeslot at which the CoAP Response (e.g. Join Response) is processed. Carried in network byte order. The seconds and fraction fields are based on the specification described in the NTP standard [RFC5905] for the Timestamp format. The seconds field accounts for seconds elapsed since the 0h on the 1 January 1900 UTC, as described by the NTP standard [RFC5905].
- o A 32-bit unsigned integer containing the number of picoseconds elapsed after the last entire second at the beginning of the timeslot at which the CoAP Response (e.g. Join Response) is processed. Carried in network byte order. Its granularity is described in the NTP standard [RFC5905].
- o Optionally, a byte string encoding a global time service URI in core-link format. The default value is 'gt'.
- o Optionally, an unsigned word lease value indicating the number of minutes of freshness of the assigned global time information. If this value is not provided the lease time is considered to be infinite. If this value is 0, a node does not refresh the global time information.

```
global_time_option = {  
    0 : bstr,           ; ASN  
    1 : uint8,         ; era counter  
    2 : uint32         ; seconds counter  
    3 : uint32         ; fraction  
    ? 4 : bstr         ; gt_service  
    ? 5 : uint16       ; gt_lease  
}
```

Name	Label	CBOR type	Reference
ASN	0	byte string	[[this document]]
era counter	1	unsigned integer (uint8)	[[this document]]
seconds counter	2	unsigned integer (uint32)	[[this document]]
fraction	3	unsigned integer (uint32)	[[this document]]
gt_service (optional)	4	byte string	[[this document]]
gt_lease (optional)	5	unsigned integer (uint16)	[[this document]]

To take into account possible leap seconds. An optional `leap_second_option` is defined by:

- o An 8-bit unsigned integer containing the action to be performed when the next leap second day is reached.
- o A 16-bit unsigned integer containing an offset in days to the beginning of the day (0 h UTC) when the next leap second must be applied. Carried in network byte order.

```
leap_second_option = {
  0 : uint8,           ; leap_indicator
  1 : uint16,          ; leap_offset
}
```

Name	Label	CBOR type	Reference
leap_indicator	0	unsigned integer (uint8)	[[this document]]
leap_offset	1	unsigned integer (uint16)	[[this document]]

The optional `leap_second_option` defines a leap second indicator, which identifies the type of correction that needs to be applied once the next leap second day is reached. The types are described in Figure 9 of the RFC5905 [RFC5905].

A `leap_offset` contains the offset in days to when the next leap second needs to be applied, following the action described in the leap second indicator.

The `global_time_option` and the `leap_second_option`, if present, SHOULD be appended following the Join Response Payload and MUST be encoded as a CBOR map objects [RFC7049].

4. Resynchronization

When a pledge receives the Join Response containing the `global_time_option`, it updates its internal absolute time clock/counter. If present, it also stores the `gt_service` link-format URI and the lease time.

After correcting a leap second or when the lease period is reached, a node MAY want to update the global time information values to keep track of the next leap second correction event or to renew its global time synchronization lease. This resynchronization is conducted through a CoAP GET Request to the JRC address and `gt_service` URI.

- o The request method is GET.
- o The type is Non-confirmable (NON).
- o The Proxy-Scheme option is set to "coap".
- o The Uri-Host option is defined by the JRC URI.
- o The Uri-Path option is set to `gt_service`.
- o The payload is empty.

The response is a CoAP Response Message with Response Code 2.05 (Content) containing the `global_time_option` as payload. The response

MAY contain a `leap_second_option` in case a leap second update is needed. Both options if present are encoded as CBOR dictionaries.

5. Leap Second handling

When a 6TiSCH node receives a global time synchronization message and this response contains a `leap_second_option`, the node MUST store the values until the leap second offset is reached.

When a leap second offset is reached, the leap second is corrected adding or subtracting a second to the last minute of the day as indicated by the `leap_indicator` field.

6. Security Considerations

The global time synchronization option is transported as part of the payload of the 6P Join Response message and secured by OSCORE (Object Security for Constrained RESTful Environments) as per [I-D.ietf-6tisch-minimal-security]. Since the JRC acts as the global time synchronization server, compromising the JRC will result in a compromise of the transported information.

A malicious 6N attacker may use a short lease time to generate high volumes of traffic in the network or even to try to deny the service to the JRC. It is up to the JRC to implement temporal blacklisting policies to avoid any DoS attack.

7. References

7.1. Normative References

- [I-D.ietf-6tisch-minimal-security] Vucinic, M., Simon, J., Pister, K., and M. Richardson, "Minimal Security Framework for 6TiSCH", draft-ietf-6tisch-minimal-security-06 (work in progress), May 2018.
- [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>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.

- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <<https://www.rfc-editor.org/info/rfc6690>>.
- [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>>.
- [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>>.

7.2. Informative 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-14 (work in progress), April 2018.
- [IEEE1588]
IEEE standard for Information Technology, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) , vol., no., pp.1-300", July 2008.

Authors' Addresses

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

Email: xvilajosana@uoc.edu

Pere Tuset
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Email: peretuset@uoc.edu

Borja Martinez
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Email: bmartinezh@uoc.edu

Jonathan Munoz
Inria
2 rue Simone Iff
Paris 75012
France

Email: jonathan.munoz@inria.fr