6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: November 25, 2017

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

May 24, 2017

6top Protocol (6P)
draft-ietf-6tisch-6top-protocol-05

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 6top Scheduling Function (SF) decides when to add/delete cells, and triggers 6P Transactions. Several SFs can be defined, each identified by a different 6top Scheduling Function Identifier (SFID). This document lists the requirements for an SF, but leaves the definition of the SF out of scope. SFs are expected to be defined in future companion specifications.

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 will expire on November 25, 2017.

Copyright Notice

Table of Contents

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 in time.
- o "TEMPORARY" appendices are there to capture current ongoing discussions, or the changelog of the document. These appendices will be removed in the final text.
- o "IANA_*" identifiers are placeholders for numbers assigned by IANA. These placeholders are to be replaced by the actual values they represent after their assignment by IANA.
- o "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, NOT part of the final text.
- o This section will be removed in the final text.

2. Introduction

All communication in a 6TiSCH network is orchestrated by a schedule [RFC7554]. This specification defines the 6top Protocol (6P), part of the 6TiSCH Operation sublayer (6top). 6P allows a node to communicate with a neighbor to add/delete TSCH cells to one another. This results in distributed schedule management in a 6TiSCH network.

```
            (R)
            / \
           /   \
        (B)-----(C)
         |       |
         |       |
        (A)     (D)
```

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

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

- If node A determines that the number of link-layer frames it is sending to 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.
- If the traffic is lower than the capacity, node A triggers a 6P Transaction with node B to delete one or more cells in the TSCH schedule of both nodes.
- Node A MAY also monitor statistics to determine whether collisions are happening on a particular cell to node B. If this feature is enabled, node A communicates with node B to add a new cell and delete the cell which suffered from collisions. This results in "relocating" the cell which suffered from collisions to a different [slotOffset,channelOffset] location in the TSCH schedule. The mechanism to handle cell relocation is out of the scope of this document and might be handled by the scheduling function (see below).

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-transaction basis.

Section 3 describes the 6TiSCH Operation Sublayer (6top). Section 4 defines the 6top Protocol (6P). Section 5 provides guidelines on how to design an SF.

3. 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 [IEEE802154-2015]. We use "802.15.4" as a short version of "IEEE Std 802.15.4" in this document.

```
                          .
   |                      .                    |
   |               higher layers               |
   +-------------------------------------------+
   |                   6top                    |
   +-------------------------------------------+
   |          IEEE Std 802.15.4 TSCH           |
   |                      .                    |
                          .
```

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

The roles of the 6top sublayer are to:

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

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

3.2. Using 6P with the Minimal 6TiSCH Configuration

6P MAY be used alongside the Minimal 6TiSCH Configuration [I-D.ietf-6tisch-minimal]. 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 length can be shorter or longer.
- o 6P allocates cells from Slotframe 1. In Figure 3, Slotframe 1 is 10 slots long, but the length can be shorter or longer.

```
                | 0    1    2    3    4   | 0    1    2    3    4   |
                +------------------------+------------------------+
    Slotframe 0 |    |    |    |    |    |    |    |    |    |    |
    5 slots long| EB |    |    |    |    | EB |    |    |    |    |
                |    |    |    |    |    |    |    |    |    |    |
                +-------------------------------------------------+

                | 0    1    2    3    4    5    6    7    8    9   |
                +-------------------------------------------------+
    Slotframe 1 |    |    |    |    |    |    |    |    |    |    |
   10 slots long|    |A->B|    |    |    |    |    |    |B->A|    |
                |    |    |    |    |    |    |    |    |    |    |
                +-------------------------------------------------+
```

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. In 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 the scope for this document.

4. 6top Protocol (6P)

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

4.1. 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 to one of its neighbors. A 6P Transaction ends when the cell(s) have been added/deleted/relocated from the schedule of both nodes, or when the 6P Transaction has failed.

The 6P messages exchanged between nodes A and B during a 6P Transaction SHOULD be exchanged on dedicated cells between A and B. If no dedicated cells are scheduled between nodes A and B, shared cells MAY be used.

Consistency between the schedules of the two neighbor nodes is of utmost importance. A loss of consistency (e.g. node A has a transmit cell to node B, but node B does not have the corresponding reception cell) can cause loss of connectivity. To verify consistency, neighbor nodes increment the "schedule generation" number of their schedule each time their schedule is modified. Neighbor nodes exchange the schedule generation number as part of each 6P Transaction to detect possible inconsistencies. This mechanism is explained in Section 4.4.7.

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 the scope of this document.

A 6P Transaction can consist of 2 or 3 steps. An SF MUST specify whether to use 2-step or 3-step transactions (or both).

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

4.1.1. 2-step 6P Transaction

Figure 4 shows an example 2-step 6P Transaction. Several elements are left out to simplify understanding.

```
          +----------+                       +----------+
          |  Node A  |                       |  Node B  |
          +----+-----+                       +-----+----+
               |                                   |
               | 6P ADD Request                    |
               |   Type         = REQUEST          |
               |   Code         = ADD              |
               |   NumCells     = 2                |
       timeout |   CellList     = [(1,2),(2,2),(3,5)]  |
         ---   |---------------------------------->|
          |    |                                   |
          |    | 6P Response                       |
          |    |   Type         = RESPONSE         |
          |    |   Code         = SUCCESS          |
          |    |   CellList     = [(2,2),(3,5)]    |
          X    |<----------------------------------|
               |                                   |
```

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.
4. When it sends the 6P ADD Request, Node A sets a timer to abort the transaction if no response has been received before the timeout.
5. The SF running on node B selects 2 out of the 3 cells in the CellList of the 6P ADD Request. Node B sends back a 6P Response to node A, indicating the cells that node B selected.
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. The schedule generation number (see Section 4.4.7) is incremented to allow inconsistency detection.

2-step transaction is used when node A selects the candidate cells.

4.1.2. 3-step 6P Transaction

Figure 5 shows an example 3-step 6P Transaction. Several elements are left out to simplify understanding.

```
           +----------+                           +----------+
           |  Node A  |                           |  Node B  |
           +----+-----+                           +-----+----+
                |                                       |
                | 6P ADD Request                        |
                |   Type         = REQUEST              |
                |   Code         = ADD                  |
                |   NumCells     = 2                    |
        timeout |   CellList     = []                   |
          ---   |-------------------------------------->|
           |    |                                       |
           |    | 6P Response                           |
           |    |   Type         = RESPONSE             |
           |    |   Code         = SUCCESS              |
           |    |   CellList     = [(1,2),(2,2),(3,5)]  | timeout
           X    |<--------------------------------------|   ---
                |                                       |    |
                | 6P Confirmation                       |    |
                |   Type         = CONFIRMATION         |    |
                |   Code         = SUCCESS              |    |
                |   CellList     = [(2,2),(3,5)]        |    |
                |-------------------------------------->|    X
                |                                       |
```

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".
3. When it sends the 6P ADD Request, Node A sets a timer to abort the transaction if no response has been received before the timeout.
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.
5. When it sends the 6P Response to node A, Node B sets a timer to abort the transaction if no response has been received before the timeout.
6. The SF running on node A selects 2 cells. Node A sends back a 6P Confirmation to node B, indicating the cells it selected.
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. The schedule generation number (see Section 4.4.7) is incremented to allow inconsistencies detection.

3-step transaction is used when node B selects the candidate cells.

4.2. Message Format

4.2.1. 6top Information Element (IE)

6P messages are carried as payload of a 802.15.4 Payload Information Element (IE) [IEEE802154-2015]. 6P messages travel over a single hop.

This document defines the "6top IE", a subtype of the IETF IE defined in [I-D.kivinen-802-15-ie], with subtype IANA_6TOP_SUBIE_ID. The length of the 6top IE content is variable.

4.2.2. Generic 6P Message Format

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

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |     Code      |      SFID     | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Other Fields...
+-+-+-+-+-+-+-+-+-
```

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 8.2.2.

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 8.2.3 lists the 6P command identifiers. The Code field contains a 6P Return Code when the 6P message is of Type RESPONSE or CONFIRMATION. Section 8.2.4 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 5.1.

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 increment by exactly one at each new 6P request issued to the same neighbor.

Schedule Generation (GEN): Schedule Generation for the transactions between node A and node B. The generation is used to ensure consistency between the schedules of the two neighbors. Section 4.4.7 details how the schedule generation is managed.

Other Fields: The list of other fields depends on the type of messages, and is detailed in Section 4.3.

4.2.3. 6P CellOptions

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

- In the 6P ADD request, the 6P CellOptions bitmap is used to specify what type of cell to add.
- In the 6P DELETE request, the 6P CellOptions bitmap is used to specify what type of cell to delete.
- In the 6P COUNT and the 6P LIST requests, the 6P CellOptions bitmap is used as a selector of a particular type of cells.
- 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 8.2.6 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.

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

CellOptions Value	cells scheduled with A that are to be selected by B when receiving a 6P message from A
TX=0,RX=0,S=0	select all cells
TX=1,RX=0,S=0	select the cells scheduled marked as RX
TX=0,RX=1,S=0	select the cells marked as TX
TX=1,RX=1,S=0	select the cells marked as TX and RX
TX=0,RX=0,S=1	select the cells marked as SHARE
TX=1,RX=0,S=1	select the cells marked as RX and SHARE
TX=0,RX=1,S=1	select the cells marked as TX and SHARE
TX=1,RX=1,S=1	select the cells marked as TX and RX and SHARE

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

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

4.2.4. 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 zero, one or more 6P Cell containers. 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:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          slotOffset           |         channelOffset         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: 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.

4.3. 6P Commands and Operations

4.3.1. 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 9 defines the format of a 6P ADD Request.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Metadata            |  CellOptions  |   NumCells    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CellList ...
+-+-+-+-+-+-+-+-+-
```

Figure 9: 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 could specify which slotframe to be used for scheduling 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 all of them. This necessarily means that, if node A needs to add multiple cells with different options, it needs to issue 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 6P Cells to be added.

Figure 10 defines the format of a 6P ADD Response and Confirmation.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CellList ...
+-+-+-+-+-+-+-+-+-
```

Figure 10: 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 as candidate cells to node B. The CellOptions field specifies the type of these cells. 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 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 in 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 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).

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

4.3.2. 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 11 defines the format of a 6P DELETE Request.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Metadata             |  CellOptions  |   NumCells    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CellList ...
   +-+-+-+-+-+-+-+-+-
```

Figure 11: 6P DELETE Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 4.3.1. Its format is same as that in 6P ADD command, but content 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 12 defines the format of a 6P DELETE Response and Confirmation.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CellList ...
   +-+-+-+-+-+-+-+-+-
```

Figure 12: 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:

- The nodes delete the cells they agree upon rather than adding them.
- All cells in the CellList MUST already be scheduled between the two nodes and must match the CellOptions field. If node A puts cells in its CellList that are not already scheduled between the two nodes and match the CellOptions field, node B replies with a RESET return code.

- o If the CellList in the 6P Request is empty, the SF on the receiving node is free to 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.

4.3.3. 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 13 defines the format of a 6P RELOCATE Request.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Metadata            |  CellOptions  |   NumCells    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Rel. CellList  ...   |Cand. CellList (Optional) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: 6P RELOCATE Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 4.3.1. Its format is same as that in 6P ADD command, but content could be different.

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 equal or greater than NumCells.

Figure 14 defines the format of a 6P RELOCATE Response and Confirmation.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CellList ...
   +-+-+-+-+-+-+-+-+-
```

Figure 14: 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'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 CELLLIST_ERR. 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 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 < 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, 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.

```
        +----------+                       +----------+
        |  Node A  |                       |  Node B  |
        +----+-----+                       +-----+----+
             |                                   |
             | 6P RELOCATE Request               |
             |   Type        = REQUEST           |
             |   Code        = RELOCATE          |
             |   NumCells    = 2                 |
             |   R.CellList  = [(1,2),(2,2)]     |
             |   C.CellList  = [(3,2),(4,2),(6,5)]  |
             |---------------------------------->| B relocates
             |                                   | (1,2)->(4,2)
             | 6P Response                       | but cannot
             |   Type        = RESPONSE          | relocate (2,2)
             |   Code        = SUCCESS           |
             |   CellList    = [(4,2)]           |
    A relocates |<----------------------------------|
    (1,2)->(4,2)|                                   |
```

Figure 15: 6P RELOCATE Example.

4.3.4. 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 16 defines the format of a 6P COUNT Request.

```
                     1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Metadata            |  CellOptions  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: 6P COUNT Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 4.3.1. Its format is same as that in 6P ADD command, but content could be different.

CellOptions: Specifies which types of cells to be counted.

Figure 17 defines the format of a 6P COUNT Response and Confirmation.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |      Code       |      SFID       | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           NumCells            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: 6P COUNT Response and Confirmation Formats.

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 4.2.3 details the use of the CellOptions field.

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

4.3.5. Listing Cells

To retrieve the 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 18 defines the format of a 6P LIST Request.

```
                     1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |      Code       |      SFID       | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Metadata            |  CellOptions  |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Offset              |          MaxNumCells          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: 6P LIST Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 4.3.1. Its format is same as that in 6P ADD command, but content could be different.

CellOptions: Specifies which types of cells to be listed.

Reserved: Set to 0.

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 returns less than MaxNumCells cells, for example if MaxNumCells cells do not fit in the frame.

Figure 19 defines the format of a 6P LIST Response and Confirmation.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| CellList ...
+-+-+-+-+-+-+-+-+-
```

Figure 19: 6P LIST Response and Confirmation Formats.

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 4.2.3

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 EOL, 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 EOL.

4.3.6. 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 20 defines the format of a 6P CLEAR Request.

```
                        1                   2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           Metadata            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: 6P CLEAR Request Format.

Metadata: Same usage as for the 6P ADD command, see Section 4.3.1. Its format is same as that in 6P ADD command, but content could be different.

Figure 21 defines the format of a 6P CLEAR Response and Confirmation.

```
                          1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Version| T | R |     Code      |     SFID      | SeqNum|  GEN  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: 6P CLEAR Response and Confirmation Formats.

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 B, and node B MUST remove all the cells scheduled with A. In a 6P CLEAR command, the generation counter GEN MUST NOT be checked. That is, its value is "don't care". In particular, even if the request contains a GEN value that would normally cause node B to detect a schedule generation mismatch, the transaction MUST NOT be aborted.

4.4. Protocol Functional Details

4.4.1. 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 receiving a 6P message with a Version number it does not implement, a node MUST reply with a 6P Response with a Return Code field set to VERSION. The Version field in the 6P Response MUST be the same as the Version field in the corresponding 6P Request. 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.

4.4.2. SFID Checking

All messages contain a 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 SFID_ERR. 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.

4.4.3. 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 RESET. A node receiving RESET code MUST abort the 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 NORES. In case the requested cells are locked, it MUST reply to that request with a 6P Response with return code BUSY. The node receiving BUSY or an NORES MAY implement a retry mechanism, defined by the SF.

4.4.4. 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. The timeout should be longer than the longest possible time it can take for the exchange to finish. The value of the 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.

4.4.5. SeqNum Mismatch

A SeqNum mismatch happens when a node receives a 6P Response or 6P Confirmation with SeqNum value different from the SeqNum value in the 6P Request. When it detects a SeqNum mismatch, the node MUST drop the packet and consider the 6P Transaction as having failed.

4.4.6. 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 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 SHOULD reply to that 6P Response with a 6P Confirmation with return code RESET. Upon receiving this 6P Confirmation, the sender of the 6P Response MUST consider the 6P Transaction as failed.

4.4.7. Generation Management

For each neighbor, a node maintains a 4-bit generation number. The generation number counts the number of transactions that have modified the schedule with the particular neighbor so far. This number is a variable internal to the node.

4.4.7.1. Incrementing the Generation Number

The generation number is incremented as a 4-bit lollipop counter. Its possible values are:

Value	Meaning
0x0	Clear or never scheduled
0x1-0x9	Lollipop Counter values
0xa-0xf	Reserved

Figure 22: Possible values of the generation number.

The generation number is set to 0 upon initialization, and after a 6P CLEAR command. The generation number is incremented by exactly 1 each time a cell with that neighbor is added/deleted/relocated from the schedule (e.g. after a successful 6P ADD, 6P DELETE or 6P RELOCATE transaction). The value rolls from 0x9 to 0x1 (i.e. not

0x0). This results in a lollipop counter with 0x0 the start value, and 0x1-0x9 the count values. Values from 0xa to 0xf are reserved and MUST NOT be used.

4.4.7.2. Setting GEN field in the 6P Message Header

Each 6P message contains a GEN field, used to indicate the current generation number of the node transmitting the message. The value of the GEN field MUST be set according to the following rules:

- o When node A sends a 6P Request or 6P Confirmation to node B, node A sets GEN to its generation number for Node B.
- o When node B sends a 6P Response to node A, node B sets GEN to its generation number for node A.

4.4.7.3. Detecting and Handling Schedule Generation Inconsistencies

Upon receiving a 6P message, a node MUST do the following checks:

- o When node B receives a 6P Request or 6P Confirmation from node A, it verifies that the value of the GEN field in the 6P message is equal to its internal generation number.
- o When node A receives a 6P Response from node B, it verifies that the value of the GEN field in the 6P message is equal to its internal generation number.

If any of these comparisons is false, the node has detected a schedule generation inconsistency.

When a schedule generation inconsistency is detected:

- o If the code of the 6P Request is different from CLEAR, the node MUST reply with error code GEN_ERR.
- o If the code of the 6P Request is CLEAR, the schedule generation inconsistency MUST be ignored.

It is up to the Scheduling Function to define the action to take when an schedule generation inconsistency is detected. The RECOMMENDED action is to issue a 6P CLEAR command.

4.4.8. Handling Error Responses

A return code marked as YES in the "Is Error" column in Figure 27 indicates an error. When a node receives a 6P Response or 6P Confirmation with such an error, it MUST consider the 6P Transaction 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.

4.5. 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.

5. Guidelines for 6top Scheduling Functions (SF)

5.1. SF Identifier (SFID)

Each SF has a 1-byte identifier. Section 8.2.5 defines the rules for applying for an SFID.

5.2. Requirements for an SF

The specification for an SF

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

- SHOULD clearly state the application domain the SF is created for.
- SHOULD contain examples which highlight normal and error scenarios.
- SHOULD contain a list of current implementations, at least during the I-D state of the document, per [RFC6982].

- o SHOULD contain a performance evaluation of the scheme, possibly through references to external documents.

- 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.

5.3. 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 6P Error Handling
- o Examples
- o Implementation Status
- o Security Considerations
- o IANA Considerations

6. 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".

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/.

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.

7. 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*. 6P benefits from the same level of security as any other Payload IE. The 6P protocol does not define its own security mechanisms. 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.

8. IANA Considerations

8.1. IETF IE Subtype '6P'

This document adds the following number to the "IEEE Std 802.15.4 IETF IE subtype IDs" registry defined by [I-D.kivinen-802-15-ie]:

Subtype	Name	Reference
IANA_6TOP_SUBIE_ID	6P	RFCXXXX

Figure 23: IETF IE Subtype '6P'.

8.2. 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.

8.2.1. 6P Version Numbers

The name of the sub-registry is "CoAP 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 24: 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 [RFC5226].

8.2.2. 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 25: 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 [RFC5226].

8.2.3. 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	CLEAR	RFCXXXX
7-254	Unassigned	
255	Reserved	

Figure 26: 6P Command Identifiers.

The IANA policy for future additions to this sub-registry is "IETF Review or IESG Approval" as described in [RFC5226].

8.2.4. 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	SUCCESS	operation succeeded	No
1	ERROR	generic error	Yes
2	EOL	end of list	No
3	RESET	critical error, reset	Yes
4	VER_ERR	unsupported 6P version	Yes
5	SFID_ERR	unsupported SFID	Yes
6	GEN_ERR	wrong schedule generation	Yes
7	BUSY	busy	Yes
8	NORES	not enough resources	Yes
9	CELLLIST_ERR	cellList error	Yes

Figure 27: 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 [RFC5226].

8.2.5. 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 entry in this sub-registry is as follows:

SFID	Name	Reference
0	Scheduling Function Zero	draft-ietf-6tisch-6top-sf0

Figure 28: 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 29. These specifications must follow the guidelines of Section 5.

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

Figure 29: SF Identifier (SFID): Registration Procedures.

8.2.6. 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 30: 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 [RFC5226].

9. References

9.1. Normative References

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

[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.

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

9.2. Informative References

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

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[I-D.ietf-6tisch-minimal] Vilajosana, X., Pister, K., and T. Watteyne, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-21 (work in progress), February 2017.

[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.

Appendix A. [TEMPORARY] Changelog

- 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
- 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
- draft-ietf-6tisch-6top-protocol-03
    * Added a reference to [I-D.kivinen-802-15-ie].
    * Added the Type field.
    * Editorial changes (figs, typos, ...)
- 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, ...)
- 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
- draft-ietf-6tisch-6top-protocol-00
    * Informational to Std track
- 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

- draft-wang-6tisch-6top-sublayer-05

   * Specifies format of IE
   * Adds token in messages to match request and response

- 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.

- 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-negociation

- 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: twatteyne@linear.com