6TiSCH
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2020

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

July 2, 2019

6TiSCH Minimal Scheduling Function (MSF)
draft-ietf-6tisch-msf-04

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, 2020.

Table of Contents

1. Introduction

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

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

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

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

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

- This specification does not include an examples section.

2. Interface to the Minimal 6TiSCH Configuration

A node implementing MSF SHOULD implement the Minimal 6TiSCH Configuration [RFC8180], which defines the "minimal cell", a single shared cell providing minimal connectivity between the nodes in the network. The MSF implementation provided in this specification is based on the implementation of the Minimal 6TiSCH Configuration. However, an implementor MAY implement MSF without implementing Minimal 6TiSCH Configuration.

MSF uses the minimal cell to exchange the following packets:

1. Enhanced Beacons (EBs), defined by [IEEE802154-2015]. These are broadcast frames.
2. Broadcast DODAG Information Objects (DIOs), defined by [RFC6550]. Unicast DIOs SHOULD NOT be sent on minimal cell.

To ensure there is enough bandwidth available on the minimal cell, a node implementing MSF SHOULD enforce some rules for limiting the traffic of broadcast frames. For example, a Trickle Timer defined in [RFC6550] MAY be applied on DIOs. However, this behvaior is implementation-specific which is out of the scope of MSF.

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

3. Autonomous Cells

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

- Autonomous Rx Cell (AutoRxCell), one cell at a [slotOffset,channelOffset] computed as a hash of the EUI64 of the

node itself (detailed next). Its cell options bits are assigned as TX=0, RX=1, SHARED=0.

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

To compute a [slotOffset,channelOffset] from an EUI64 address, nodes MUST use the hash function SAX [SAX-DASFAA]. The coordinates are computed to distribute the cells across all 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, NUM_CH_OFFSET)

The second input parameter defines the maximum return value of the hash function. Other optional parameters defined in SAX determine the performance of SAX hash function. Those parameters could be broadcasted in EB frame or pre-configured. For interoperability purposes, an example how the hash function is implemented is detailed in Appendix B.

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

- Add an AutoTxCell to the layer 2 source address which is indicated in a frame when:
  - there is no 6P negotiated Tx cell in schedule for that frame to transmit, and
  - the frame is used for protocol management purposes , such as Join Request/Response, 6P Request/Response and any link-local communication for RPL routing control.
- Remove an AutoTxCell when:
  - there is no frame for protocol management purposes to transmit, or
  - there is at least one 6P negotiated Tx cell in the schedule to transmit a management purpose frame.
- The AutoRxCell MUST always remain scheduled.
- 6P CLEAR MUST NOT erase any autonomous cells.

Because of hash collisions, there will be cases when the AutoTxCell and AutoRxCell are scheduled at the same slot offset and/or channel offset. In such cases, AutoTxCell always take precedence over AutoRxCell. In case of conflicting with a negotiated cell, autonomous cells take precedence over negotiated cell. However, when the Slotframe 0, 1 and 2 use the same length value, it is possible for negoitated cell to avoid the collision with AutoRxCell.

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

4.1. Start State

A node implementing MSF SHOULD implement the Minimal Security Framework for 6TiSCH [I-D.ietf-6tisch-minimal-security]. As a corollary, this means that a pledge, before being switched on, may be pre-configured with the Pre-Shared Key (PSK) for joining, as well as any other configuration detailed in ([I-D.ietf-6tisch-minimal-security]). This is not needed if the node implements a security solution not based on PSKs, such as ([I-D.ietf-6tisch-dtsecurity-zerotouch-join]).

4.2. Step 1 - Choosing Frequency

When switched on, the pledge 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 Cells for the Join Process

After selected a JP, a node generates a Join Request and installs an AutoTxCell to the JP. A Join Request is then sent by the pledge to its JP over the AutoTxCell. The AutoTxCell is removed by the pledge when the Join Request is sent out. The JP receives the Join Request through its AutoRxCell. Then it forwards the Join Request to the JRC, possibly over multiple hops, over the 6P negotiated Tx cell. Similarly, the JRC sends the Join Response to the JP, possibly over multiple hops, over the 6P negotiated Tx cell. When JP received the Join Response from the JRC, it installs an AutoTxCell to the pledge and sends that Join Response to the pledge over AutoTxCell. The AutoTxCell is removed by the JP when the Join Response is sent out. The pledge receives the Join Response from its AutoRxCell, thereby learns the keying material used in the network, as well as other configurations, and becomes a "joined node".

4.5. Step 4 - Acquiring a RPL rank

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

4.6. Step 5 - Setting up first Tx and Rx negotiated Cells

After selected a preferred parent, the joined node MUST generate a 6P ADD Request and install an AutoTxCell to that parent. The 6P ADD Request is sent out through the AutoTxCell with the following fields:

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

The joined node removes the AutoTxCell to parent when the 6P command is send out. Its parent receives the 6P ADD Request from its AutoRxCell. Then it generates a 6P ADD Response and installs an AutoTxCell to the joined node. When the parent sends out the 6P ADD Response, it MUST remove that AutoTxCell. The joined node receives

the 6P ADD Response from its AutoRxCell and completes the 6P transcation. In case the 6P ADD transaction failed, the node MUST issue another 6P ADD command and repeat until the Tx cell is installed to the parent.

After the first negotiated Tx Cell is installed, the joined node SHOULD send another 6P ADD Request with the following fields:

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

The process to install a negotiated Rx cell is the similar with the process to install a negotiated Tx cell. The only difference is that the 6P ADD Request is sent on the neogtiated Tx cell installed previously, instead of the AutoTxCell.

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. End State

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

- it is synchronized to the network
- it is using the link-layer keying material it learned through the secure joining process
- it has identified its preferred routing parent
- it has one one AutRxCell
- it has one negotiated Tx cell and one negotiated Rx cell to its parent
- it starts to send DIOs, potentially serving as a router for other nodes' traffic
- it starts to send EBs, potentially serving as a JP for new pledge

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:

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

Those cells are called 'negotiated cells' as they are scheduled through 6P, negotiated with their parents. Without specific declaring, all cells mentioned in this section are negotiated cells and they are installed at Slotframe 2.

5.1. Adapting to Traffic

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

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

- 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 negotiated cells it has scheduled with it, the node issues a 6P ADD command to its preferred parent to add cells to the TSCH schedule.
- If the traffic is lower than the capacity, the node issues a 6P DELETE command to its preferred parent to delete cells from the TSCH schedule.

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

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

NumCellsUsed: Counts the number of negotiated cells that have been used. This counter is initialized at 0. NumCellsUsed is incremented by exactly 1 when, during a negotiated cell to the 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. The counter increments regardless of whether the frame is a valid IEEE802.15.4 frame or not.

Both NumCellsElapsed and NumCellsUsed counters can be used to negotiated cells with cell option TX=1 or Rx=1. All the frames used

for increasing/decreasing the counters MUST be encrypted or decryptable with the key get from joining process.

Implementors MAY choose to create the same counters for each neighbor, and add them as additional statistics in the neighbor table.

The counters are used as follows:

1. Both NumCellsElapsed and NumCellsUsed are initialized to 0 when the node boots.
2. When the value of NumCellsElapsed reaches MAX_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 NumCellsElapsed and NumCellsUsed to 0 and go to step 2.

The value of MAX_NUMCELLS is chosen according to the traffic type of the network. Generally speaking, the larger the value MAX_NUMCELLS is, the more accurate the cell usage is calulated. The 6P traffic overhead using a larger value of MAX_NUMCELLS could be reduced as well. Meanwhile, the latency won't increaase much by using a larger value of MAX_NUMCELLS for periodic traffic type. For burst traffic type, larger value of MAX_NUMCELLS indeed introduces higher latency. The latency caused by slight changes of traffic load can be absolved by the additional scheduled cells. In this sense, MSF is a scheduling function trading latency with energy by scheduling more cells than needed. It is recommended to set MAX_NUMCELLS value at least 4 times than the maximum link traffic load of the network in packets per slotframe. For example, a 2 packets/slotframe traffic load results an average 4 cells scheduled, using the value of double number of scheduled cells (which is 8) as MAX_NUMCELLS gives a good resolution on cell usage calculation.

5.2. Switching Parent

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

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

1. if there is negotiated cell conflicted with the AutoUpCells to be installed, the node MUST issue a 6P RELOCATE command to relocate the conflicted cell
2. if there is no conflicted cell, the node installs the AutoUpCells to its new parent
3. the node counts the number of negotiated cells it has per slotframe to the old preferred parent
4. the node triggers one or more 6P ADD commands to schedule the same number of negotiated cells to the new preferred parent
5. 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 negotiated cell at the same [slotOffset,channelOffset] location in the TSCH schedule. In that case, data exchanged by the two pairs may collide on that cell. We call this case a "schedule collision".

The node MUST maintain the following counters for each managed unicast 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 negotiated cell in the schedule.

Since both NumTx and NumTxAck are initialized to 0, we necessarily have NumTxAck <= NumTx. We call Packet Delivery Ratio (PDR) the ratio NumTxAck/NumTx; and represent it as a percentage. A cell with PDR=50% means that half of the frames transmitted are not acknowledged (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 than there are collisions on that cell.

Every HOUSEKEEPINGCOLLISION_PERIOD, the node executes the following steps:

1. It computes, for each managed unicast 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 any other cell, it compares its PDR against that of the cell with the highest PDR. If the difference is 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:

- The CellList SHOULD contain 5 or more cells.
- Each cell in the CellList MUST have a different slotOffset value.
- For each cell in the CellList, the node MUST NOT have any scheduled cell on the same slotOffset.
- The slotOffset value of any cell in the CellList MUST NOT be the same as the slotOffset of the minimal cell (slotOffset=0).

- The slotOffset of a cell in the CellList SHOULD be randomly and uniformly chosen among all the slotOffset values that satisfy the restrictions above.
- The channelOffset of a cell in the CellList SHOULD be randomly and uniformly chosen in [0..numFrequencies], where numFrequencies represents the number of frequencies a node can communicate on.

9. 6P Timeout Value

It is calculated for the worst case that a 6P response is received, which means the 6P response is sent out successfully at the very latest retransmission. And for each retransmission, it backs-off with largest value. Hence the 6P timeout value is calculated as ((2^MAXBE)-1)*SLOTFRAME_LENGTH, where:

- MAXBE is the maximum backoff exponent used
- SLOTFRAME_LENGTH represents the length of slotframe

10. Rule for Ordering Cells

Cells are ordered slotOffset first, channelOffset second.

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

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

11. Meaning of the Metadata Field

The Metadata field is not used by MSF.

12. 6P Error Handling

Section 6.2.4 of [RFC8480] lists the 6P Return Codes. Figure 1 lists the same error codes, and the behavior a node implementing MSF SHOULD follow.

Code	RECOMMENDED behavior
RC_SUCCESS	nothing
RC_EOL	nothing
RC_ERR	quarantine
RC_RESET	quarantine
RC_ERR_VERSION	quarantine
RC_ERR_SFID	quarantine
RC_ERR_SEQNUM	clear
RC_ERR_CELLLIST	clear
RC_ERR_BUSY	waitretry
RC_ERR_LOCKED	waitretry

Figure 1: Recommended behavior for each 6P Error Code.

The meaning of each behavior from Figure 1 is:

nothing: Indicates that this Return Code is not an error. No error handling behavior is triggered.

clear: Abort the 6P Transaction. Issue a 6P CLEAR command to that neighbor (this command may fail at the link layer). Remove all cells scheduled with that neighbor from the local schedule. 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
NUM_CH_OFFSET	16
LIM_NUMCELLSUSED_HIGH	75 %
LIM_NUMCELLSUSED_LOW	25 %
HOUSEKEEPINGCOLLISION_PERIOD	1 min
RELOCATE_PDRTHRES	50 %
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
NumCellsElapsed	1 byte
NumCellsUsed	1 byte
NumTx	1 byte
NumTxAck	1 byte

Figure 3: MSF Statistics and their RECOMMENDED width.

16. Security Considerations

MSF defines a series of "rules" for the node to follow. It triggers several actions, that are carried out by the protocols defined in the following specifications: the Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration [RFC8180], the 6TiSCH Operation Sublayer Protocol (6P) [RFC8480], and the 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 [RFC8480]:

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-dtsecurity-zerotouch-join]
Richardson, M., "6tisch Zero-Touch Secure Join protocol", draft-ietf-6tisch-dtsecurity-zerotouch-join-03 (work in progress), October 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-11 (work in progress), June 2019.

[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>.

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

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

18.2. Informative References

[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) Olaf Landsiedel (Chalmers University, olafl@chalmers.se) Yasuyuki Tanaka (Inria-Paris, yasuyuki.tanaka@inria.fr)

Appendix B. Example of Implementation of SAX hash function

For the consideration of interoperability, this section provides an example of implemention SAX hash function [SAX-DASFAA]. The input parameters of the function are:

- T, which is the hashing table length
- c, which is the characters of string s, to be hashed

In MSF, the T is replaced by the length slotframe 1. String s is replaced by the mote EUI64 address. The characters of the string c0, c1, ..., c7 are the 8 bytes of EUI64 address.

The SAX hash function requires shift operation which is defined as follow:

- L_shift(v,b), which refers to left shift variable v by b bits
- R_shift(v,b), which refers to right shift variable v by b bits

The steps to calculate the hash value of SAX hash function are:

1. initialize variable h to h0 and variable i to 0, where h is the intermediate hash value and i is the index of the bytes of EUI64 address
2. sum the value of L_shift(h,l_bit), R_shift(h,r_bit) and ci
3. calculate the result of exclusive or between the sum value in Step 2 and h
4. modulo the result of Step 3 by T
5. assign the result of Step 4 to h
6. increase i by 1
7. repeat Step2 to Step 6 until i reaches to 8
8. assign the result of Step 5 to h

The value of variable h the hash value of SAX hash function.

For interoperability purposes, the values of h0, l_bit and r_bit in Step 1 and 2 are configured as:

- h0 = 0
- l_bit = 0
- r_bit = 1

The appropriate values of l_bit and r_bit could vary depending on the the set of motes' EUI64 address. How to find those values is out of the scope of this specification.

Authors' Addresses

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

Email: tengfei.chang@inria.fr

Malisa Vucinic
Inria
2 rue Simone Iff
Paris 75012
France

Email: malisa.vucinic@inria.fr

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
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