Stream: Internet Engineering Task Force (IETF)
RFC: 9033
Category: Standards Track
Published: May 2021
ISSN: 2070-1721
Authors: T. Chang, Ed. *Inria*
M. Vučinić *Inria*
X. Vilajosana *Universitat Oberta de Catalunya*
S. Duquennoy *RISE SICS*
D. Dujovne *Universidad Diego Portales*

RFC 9033
6TiSCH Minimal Scheduling Function (MSF)

Abstract

This specification defines the "IPv6 over the TSCH mode of IEEE 802.15.4" (6TiSCH) Minimal Scheduling Function (MSF). This Scheduling Function describes both the behavior of a node when joining the network and how the communication schedule is managed in a distributed fashion. MSF is built upon the 6TiSCH Operation Sublayer Protocol (6P) and the minimal security framework for 6TiSCH.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9033.

Copyright Notice

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 and 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: "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration" [RFC8180], "6TiSCH Operation Sublayer (6top) Protocol (6P)" [RFC8480], and "Constrained Join Protocol (CoJP) for 6TiSCH" [RFC9031].

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 six steps described in Section 4. The end state of the join process is that the node is synchronized to the network, has mutually authenticated with the network, has identified a routing parent, and has scheduled one negotiated Tx cell (defined in Section 5.1) to/from its routing parent. After the join process, the node can continuously add, delete, and relocate cells as described in Section 5. It does so for three reasons: to match the link-layer resources to the traffic, to handle changing parent, and to handle a schedule collision.

MSF works closely with the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), specifically the routing parent defined in [RFC6550]. This specification only describes how MSF works with the routing parent; this parent is referred to as the "selected parent". The activity of MSF towards the single routing parent is called a "MSF session". Though the performance of MSF

is evaluated only when the "selected parent" represents the node's preferred parent, there should be no restrictions to use multiple MSF sessions, one per parent. The distribution of traffic over multiple parents is a routing decision that is out of scope for MSF.

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

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

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

1.1. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Related Documents

This specification uses messages and variables defined in IEEE Std 802.15.4-2015 [IEEE802154]. It is expected that those resources will remain in the future versions of IEEE Std 802.15.4; in which case, this specification also applies to those future versions. In the remainder of the document, we use [IEEE802154] to refer to IEEE Std 802.15.4-2015 as well as future versions of IEEE Std 802.15.4 that remain compatible.

2. Interface to the Minimal 6TiSCH Configuration

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

A node implementing MSF **SHOULD** implement the minimal 6TiSCH configuration [RFC8180], which defines the "minimal cell", a single shared cell providing minimal connectivity between the nodes in the network. The MSF implementation provided in this specification is based on the implementation of the minimal 6TiSCH configuration. However, an implementor **MAY** implement MSF based on other specifications as long as the specification defines a way to advertise the Enhanced Beacons (EBs) and DODAG Information Objects (DIOs) among the network.

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

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

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

3. Autonomous Cells

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

Autonomous Rx Cell (AutoRxCell): One cell at a [slotOffset,channelOffset] computed as a hash of the 64-bit Extended Unique Identifier (EUI-64) 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 EUI-64 destination address in the unicast frame to be transmitted (detailed in Section 4.4). Its cell options bits are assigned as TX=1, RX=0, SHARED=1.

To compute a [slotOffset,channelOffset] from an EUI-64 address, nodes **MUST** use the hash function SAX as defined in Section 2 of [SAX-DASFAA] with consistent input parameters, for example, those defined in Appendix A. The coordinates are computed to distribute the cells across all channel offsets, and all but the first slot offset of Slotframe 1. The first time offset is skipped to avoid colliding with the minimal cell in Slotframe 0. The slot coordinates derived from a given EUI-64 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 broadcast in an EB frame or preconfigured. For interoperability purposes, Appendix A provides the reference values of those parameters.

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

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

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

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

4. Node Behavior at Boot

This section details the behavior the node **SHOULD** follow from the moment it is switched on until it has successfully joined the network. Alternative behaviors may be involved, for example, when alternative security solutions are used for the network. Section 4.1 details the start state; Section 4.8 details the end state. The other sections detail the six steps of the joining process. We use the term "pledge" and "joined node", as defined in [RFC9031].

4.1. Start State

A node implementing MSF **SHOULD** implement the Constrained Join Protocol (CoJP) for 6TiSCH [RFC9031]. As a corollary, this means that a pledge, before being switched on, may be preconfigured with the Pre-Shared Key (PSK) for joining, as well as any other configuration detailed in [RFC9031]. This is not necessary if the node implements a security solution that is not based on PSKs, such as [ZEROTOUCH-JOIN].

4.2. Step 1 - Choosing Frequency

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

4.3. Step 2 - Receiving EBs

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

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

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

During this step, the pledge only gets synchronized when it has received enough EB from the network it wishes to join. How to decide whether an EB originates from a node from the network it wishes to join is implementation-specific, but **MAY** involve filtering EBs by the PANID field it contains, the presence and contents of the Information Element (IE) defined in [RFC9032], or the key used to authenticate it.

The decision of which neighbor to use as a JP is implementation-specific and is discussed in [RFC9031].

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

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

When 6LoWPAN Neighbor Discovery (ND) [RFC8505] is implemented, the unicast packets used by ND are sent on the AutoTxCell. The specific process how the ND works during the join process is detailed in [RFC9030].

4.5. Step 4 - Acquiring a RPL Rank

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

4.6. Step 5 - Setting up First Tx Negotiated Cells

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

CellOptions: Set to TX=1, RX=0, SHARED=0.

NumCells: Set to 1.

CellList: At least 5 cells, chosen according to Section 8.

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

4.7. Step 6 - Sending EBs and DIOs

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

4.8. End State

At the end state of the joining process, a new node:

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

5. Rules for Adding and Deleting Cells

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

- to match the link-layer resources to the traffic between the node and the selected parent (Section 5.1),
- to handle switching the parent (Section 5.2), or

- to handle a schedule collision (Section 5.3).

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

5.1. Adapting to Traffic

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

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

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

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

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

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

- The node sends a frame to the parent. The counter increments regardless of whether a link-layer acknowledgment was received or not.
- The node receives a valid frame from the parent. The counter increments only when a valid frame per [IEEE802154] is received by the node from its parent.

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

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

- NumCellsElapsed is incremented by exactly 1 when the current cell is AutoRxCell.

- NumCellsUsed is incremented by exactly 1 when the node receives a frame from the selected parent on AutoRxCell.

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

The counters are used as follows:

1. Both NumCellsElapsed and NumCellsUsed are initialized to 0 when the node boots.
2. When the value of NumCellsElapsed reaches MAX_NUM_CELLS:
   - If NumCellsUsed is greater than LIM_NUMCELLSUSED_HIGH, trigger 6P to add a single cell to the selected parent.
   - If NumCellsUsed is less than LIM_NUMCELLSUSED_LOW, trigger 6P to remove a single cell to the selected parent.
   - Reset both NumCellsElapsed and NumCellsUsed to 0 and restart #2.

The value of MAX_NUM_CELLS is chosen according to the traffic type of the network. Generally speaking, the larger the value MAX_NUM_CELLS is, the more accurately the cell usage is calculated. By using a larger value of MAX_NUM_CELLS, the 6P traffic overhead could be reduced as well. Meanwhile, the latency won't increase much by using a larger value of MAX_NUM_CELLS for periodic traffic type. For bursty traffic, a larger value of MAX_NUM_CELLS indeed introduces higher latency. The latency caused by slight changes of traffic load can be alleviated by the additional scheduled cells. In this sense, MSF is a Scheduling Function that trades latency with energy by scheduling more cells than needed. Setting MAX_NUM_CELLS to a value at least four times the recent maximum number of cells used in a slotframe is **RECOMMENDED**. For example, a two packets/slotframe traffic load results in an average of four cells scheduled (two cells are used), using at least the value of double the number of scheduled cells (which is eight) as MAX_NUM_CELLS gives a good resolution on the cell usage calculation.

In the case that a node has booted or has disappeared from the network, the cell reserved at the selected parent may be kept in the schedule forever. A cleanup mechanism **MUST** be provided to resolve this issue. The cleanup mechanism is implementation-specific. The goal is to confirm that those negotiated cells are not used anymore by the associated neighbors and remove them from the schedule.

5.2. Switching Parent

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

As part of its normal operation, RPL can have a node switch parent. The procedure for switching from the old parent to the new parent is the following:

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

The type of negotiated cell that should be installed first depends on which traffic has the higher priority, upstream or downstream, which is application-specific and out of scope of MSF.

5.3. Handling Schedule Collisions

A node implementing MSF **SHOULD** implement the behavior described in this section. Other algorithms for handling schedule collisions can be an alternative to the algorithm proposed in this section.

Since scheduling is entirely distributed, there is a nonzero probability that two pairs of nearby neighbor nodes schedule a negotiated cell at the same [slotOffset,channelOffset] location in the TSCH schedule. In that case, data exchanged by the two pairs may collide on that cell. We call this case a "schedule collision".

The node **MUST** maintain the following counters for each negotiated Tx cell to the selected parent:

NumTx: Counts the number of transmission attempts on that cell. Each time the node attempts to transmit a frame on that cell, NumTx is incremented by exactly 1.

NumTxAck: Counts the number of successful transmission attempts on that cell. Each time the node receives an acknowledgment for a transmission attempt, NumTxAck is incremented by exactly 1.

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

Each time the node switches parent (or during the join process when the node selects a parent for the first time), both NumTx and NumTxAck **MUST** be reset to 0. They increment over time, as the schedule is executed, and the node sends frames to that parent. When NumTx reaches MAX_NUMTX, both NumTx and NumTxAck **MUST** be divided by 2. MAX_NUMTX needs to be a power of two to avoid division error. For example, when MAX_NUMTX is set to 256, and NumTx=255 and NumTxAck=127, the counters become NumTx=128 and NumTxAck=64 if one frame is sent to the parent with an acknowledgment received. This operation does not change the value of the PDR but allows the counters to keep incrementing. The value of MAX_NUMTX is implementation-specific.

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

Every HOUSEKEEPINGCOLLISION_PERIOD, the node executes the following steps:

1. It computes, for each negotiated Tx cell with the parent (not for the autonomous cell), that cell's PDR.

2. Any cell that hasn't yet had NumTx divided by 2 since it was last reset is skipped in steps 3 and 4. This avoids triggering cell relocation when the values of NumTx and NumTxAck are not statistically significant yet.
3. It identifies the cell with the highest PDR.
4. For any other cell, it compares its PDR against that of the cell with the highest PDR. If the subtraction difference between the PDR of the cell and the highest PDR is larger than RELOCATE_PDRTHRES, it triggers the relocation of that cell using a 6P RELOCATE command.

The RELOCATION for negotiated Rx cells is not supported by MSF.

6. 6P SIGNAL Command

The 6P SIGNAL command is not used by MSF.

7. Scheduling Function Identifier

The Scheduling Function Identifier (SFID) of MSF is 0. How the value of 0 was chosen is described in Section 17.

8. Rules for CellList

MSF uses two-step 6P Transactions exclusively. 6P Transactions are only initiated by a node towards its parent. As a result, the cells to put in the CellList of a 6P ADD command, and in the candidate CellList of a RELOCATE command, are chosen by the node initiating the 6P Transaction. In both cases, the same rules apply:

- The CellList is **RECOMMENDED** to have five 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 from [0..numFrequencies], where numFrequencies represents the number of frequencies a node can communicate on.

As a consequence of random cell selection, there is a nonzero chance that nodes in the vicinity have installed cells with same slotOffset and channelOffset. An implementer **MAY** implement a strategy to monitor the candidate cells before adding them in CellList to avoid collision. For example, a node **MAY** maintain a candidate cell pool for the CellList. The candidate cells in the pool are preconfigured as Rx cells to promiscuously listen to detect transmissions on those cells. If transmissions that rely on [IEEE802154] are observed on one cell over multiple iterations of

the schedule, that cell is probably used by a TSCH neighbor. It is moved out from the pool, and a new cell is selected as a candidate cell. The cells in CellList are picked from the candidate pool directly when required.

9. 6P Timeout Value

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

- MAXBE, defined in [IEEE802154], is the maximum backoff exponent used.
- MAXRETRIES, defined in [IEEE802154], is the maximum retransmission times.
- SLOTFRAME_LENGTH represents the length of slotframe.

10. Rule for Ordering Cells

Cells are ordered by slotOffset first, channelOffset second.

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

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

11. Meaning of the Metadata Field

The Metadata field is not used by MSF.

12. 6P Error Handling

Section 6.2.4 of [RFC8480] lists the 6P return codes. Table 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

Code	RECOMMENDED Behavior
RC_ERR_SFID	quarantine
RC_ERR_SEQNUM	clear
RC_ERR_CELLLIST	clear
RC_ERR_BUSY	waitretry
RC_ERR_LOCKED	waitretry

Table 1: Recommended Behavior for Each 6P Error Code

The meaning of each behavior from Table 1 is:

nothing: Indicates that this return code is not an error. No error handling behavior is triggered.

clear: Abort the 6P Transaction. Issue a 6P CLEAR command to that neighbor (this command may fail at the link layer). Remove all cells scheduled with that neighbor from the local schedule.

quarantine: Same behavior as for "clear". In addition, remove the node from the neighbor and routing tables. Place the node's identifier in a quarantine list for QUARANTINE_DURATION. When in quarantine, drop all frames received from that node.

waitretry: Abort the 6P Transaction. Wait for a duration randomly and uniformly chosen from [WAIT_DURATION_MIN,WAIT_DURATION_MAX]. Retry the same transaction.

13. Schedule Inconsistency Handling

The behavior when schedule inconsistency is detected is explained in Table 1, for 6P return code RC_ERR_SEQNUM.

14. MSF Constants

Table 2 lists MSF constants and their **RECOMMENDED** values.

Name	RECOMMENDED value
SLOTFRAME_LENGTH	101 slots
NUM_CH_OFFSET	16
MAX_NUM_CELLS	100
LIM_NUMCELLSUSED_HIGH	75

Name	RECOMMENDED value
LIM_NUMCELLSUSED_LOW	25
MAX_NUMTX	256
HOUSEKEEPINGCOLLISION_PERIOD	1 min
RELOCATE_PDRTHRES	50 %
QUARANTINE_DURATION	5 min
WAIT_DURATION_MIN	30 s
WAIT_DURATION_MAX	60 s

*Table 2: MSF Constants and Their **RECOMMENDED** Values*

15. MSF Statistics

Table 3 lists MSF statistics and their **RECOMMENDED** widths.

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

*Table 3: MSF Statistics and Their **RECOMMENDED** Widths*

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: "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration" [RFC8180], "6TiSCH Operation Sublayer (6top) Protocol (6P)" [RFC8480], and "Constrained Join Protocol (CoJP) for 6TiSCH" [RFC9031]. Confidentiality and authentication of MSF control and data traffic are provided by these specifications whose security considerations continue to apply to MSF. In particular, MSF does not define a new protocol or packet format.

MSF uses autonomous cells for initial bootstrap and the transport of join traffic. Autonomous cells are computed as a hash of nodes' EUI-64 addresses. This makes the coordinates of autonomous cell an easy target for an attacker, as EUI-64 addresses are visible on the wire and

are not encrypted by the link-layer security mechanism. With the coordinates of autonomous cells available, the attacker can launch a selective jamming attack against any node's AutoRxCell. If the attacker targets a node acting as a JP, it can prevent pledges from using that JP to join the network. The pledge detects such a situation through the absence of a link-layer acknowledgment for its Join Request. As it is expected that each pledge will have more than one JP available to join the network, one available countermeasure for the pledge is to pseudorandomly select a new JP when the link to the previous JP appears bad. Such a strategy alleviates the issue of the attacker randomly jamming to disturb the network but does not help in the case the attacker is targeting a particular pledge. In that case, the attacker can jam the AutoRxCell of the pledge in order to prevent it from receiving the join response. This situation should be detected through the absence of a particular node from the network and handled by the network administrator through out-of-band means.

MSF adapts to traffic containing packets from the IP layer. It is possible that the IP packet has a nonzero DSCP (Differentiated Services Code Point) [RFC2474] value in its IPv6 header. The decision how to handle that packet belongs to the upper layer and is out of scope of MSF. As long as the decision is made to hand over to MAC layer to transmit, MSF will take that packet into account when adapting to traffic.

Note that nonzero DSCP values may imply that the traffic originated at unauthenticated pledges (see [RFC9031]). The implementation at the IPv6 layer **SHOULD** rate limit this join traffic before it is passed to the 6top sublayer where MSF can observe it. If there is no rate limit for join traffic, intermediate nodes in the 6TiSCH network may be prone to a resource exhaustion attack, with the attacker injecting unauthenticated traffic from the network edge. The assumption is that the rate-limiting function is aware of the available bandwidth in the 6top Layer 3 bundle(s) towards a next hop, not directly from MSF, but from an interaction with the 6top sublayer that ultimately manages the bundles under MSF's guidance. How this rate limit is implemented is out of scope of MSF.

17. IANA Considerations

17.1. MSF Scheduling Function Identifiers

This document adds the following number to the "6P Scheduling Function Identifiers" subregistry, part of the "IPv6 Over the TSCH Mode of IEEE 802.15.4 (6TiSCH)" registry, as defined by [RFC8480]:

SFID	Name	Reference
0	Minimal Scheduling Function (MSF)	RFC 9033

Table 4: New SFID in the "6P Scheduling Function Identifiers" Subregistry

The SFID was chosen from the range 0-127, which has the registration procedure of IETF Review or IESG Approval [RFC8126].

18. References

18.1. Normative References

[IEEE802154] IEEE, "IEEE Standard for Low-Rate Wireless Networks", IEEE Standard 802.15.4-2015, DOI 10.1109/IEEESTD.2016.7460875, April 2016, <https://ieeexplore.ieee.org/document/7460875>.

[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>.

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

[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>.

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

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

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

[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>.

[RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <https://www.rfc-editor.org/info/rfc9030>.

[RFC9031] Vučinić, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/info/rfc9031>.

[RFC9032] Dujovne, D., Ed. and M. Richardson, "Encapsulation of 6TiSCH Join and Enrollment Information Elements", RFC 9032, DOI 10.17487/RFC9032, May 2021, <https://www.rfc-editor.org/info/rfc9032>.

[SAX-DASFAA] Ramakrishna, M.V. and J. Zobel, "Performance in Practice of String Hashing Functions", DASFAA, DOI 10.1142/9789812819536_0023, 1997, <https://doi.org/10.1142/9789812819536_0023>.

18.2. Informative References

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <https://www.rfc-editor.org/info/rfc6206>.

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

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

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

Appendix A. Example Implementation of the SAX Hash Function

To support interoperability, this section provides an example implementation of the SAX hash function [SAX-DASFAA]. The input parameters of the function are:

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

In MSF, the T is replaced by the length of slotframe 1. String s is replaced by the node EUI-64 address. The characters of the string, c0 through c7, are the eight bytes of the EUI-64 address.

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

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

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

1. Initialize variable h, which is the intermediate hash value, to h0 and variable i, which is the index of the bytes of the EUI-64 address, to 0.
2. Sum the value of L_shift(h,l_bit), R_shift(h,r_bit), and ci.
3. Calculate the result of the 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 Step 2 to Step 6 until i reaches to 8.

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

The values of h0, l_bit, and r_bit in Step 1 and Step 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 set of nodes' EUI-64 address. How to find those values is out of the scope of this specification.

Contributors

Beshr Al Nahas
Chalmers University
Email: beshr@chalmers.se

Olaf Landsiedel
Chalmers University
Email: olafl@chalmers.se

Yasuyuki Tanaka
Toshiba
Email: yatch1.tanaka@toshiba.co.jp

Authors' Addresses

Tengfei Chang (EDITOR)
Inria
2 rue Simone Iff
75012 Paris
France
Email: tengfei.chang@gmail.com

Mališa Vučinić
Inria
2 rue Simone Iff
75012 Paris
France
Email: malisa.vucinic@inria.fr

Xavier Vilajosana
Universitat Oberta de Catalunya
156 Rambla Poblenou
08018 Barcelona Catalonia
Spain
Email: xvilajosana@uoc.edu

Simon Duquennoy
RISE SICS
Isafjordsgatan 22
SE-164 29 Kista
Sweden
Email: simon.duquennoy@gmail.com

Diego Dujovne
Universidad Diego Portales
Escuela de Informática y Telecomunicaciones
Av. Ejército 441
Santiago
Región Metropolitana
Chile
Phone: +56 (2) 676-8121
Email: diego.dujovne@mail.udp.cl