Pushing 6TiSCH Minimal Scheduling Function (MSF) to the Limits

David Hauweele*, Remous-Aris Koutsiamanis†, Bruno Quoitin* and Georgios Z. Papadopoulos†

* University of Mons (UMONS), Belgium, Email: {first.last}@umons.ac.be

† IMT Atlantique, IRISA, France, Email: {first.last}@imt-atlantique.fr

***Abstract*—IEEE Std 802.15.4-2015 Time Slotted Channel Hopping (TSCH) is the *de facto* Medium Access Control (MAC) mechanism for industrial applications. It renders communications more resilient to interference by spreading them over the time (time slotted) and the frequency (channel hopping) domains. The 6TiSCH architecture bases itself on this new MAC layer to enable high reliability communication in Wireless Sensor Networks (WSNs). In particular, it manages the construction of a distributed communication schedule that continuously adapts to changes in the network. In this paper, we first provide a thorough description of the 6TiSCH architecture, the 6TiSCH Operation Sublayer (6top), and the Minimal Scheduling Function (MSF). We then study its behavior and reactivity from low to high traffic rates by employing the python-based 6TiSCH simulator. Our performance evaluation results demonstrate that the convergence pattern of MSF is the root cause of the majority of packet losses observed in the network. We also show that MSF is subject to over-provisioning of the network resources, especially in the case of varying traffic load.**

***Index Terms*—Internet of Things, IoT, Industrial IoT, IIoT, IEEE Std 802.15.4-2015, TSCH, 6TiSCH, 6top, 6P, MSF**

I. Introduction

Industry 4.0 is currently the focus of major development efforts aiming at making manufacturing processes more flexible, more autonomous and more economical to operate. One way in which this is being pursued is by deploying Internet of Things (IoT) technologies for the purpose of connecting management, reporting, sensing, and control interfaces. The IoT encompasses technologies which support the large-scale deployment of and communication between small, inexpensive, but often severely constrained devices. Indeed, although such devices allow great flexibility, easy mobility, and interoperability, the hardware used is by necessity limited in CPU performance, memory storage, radio communication range and energy consumption.

To compensate for these shortcomings and to allow industrial use of these devices, a set of technologies have been developed for the Industrial Internet of Things (IIoT). This context poses requirements of very high reliability, low latency, and low jitter on data transmission. Additionally, it mandates that these requirements need to be met while maintaining high energy efficiency and low CPU and memory overhead.

Recently, wireless technologies have been used with good results in terms of reliability and latency [1]. However, because of strict constraints on available network resources and required energy efficiency, assumptions are made about the characteristics of the served traffic in such networks, such as constant rate. However, replacing legacy, wire-based infrastructure requires the ability to quickly adapt to changing traffic.

To this end, the IETF has introduced a still work-in-progress functionality known as the 6TiSCH Minimal Scheduling Function, which allows the negotiation and reservation of network resources in a scheduled / deterministic access wireless network in an on-demand manner. In this paper, we describe this functionality in detail and assess its performance. We proceed by studying the behavior and reactivity of 6TiSCH MSF on simple topologies with varying traffic load. From these observations we illustrate shortcomings that it's use brings to the IIoT use case.

The rest of this paper is organized as follows. Section II presents a description of the 6TiSCH architecture. Then we describe in Section III the Minimal Scheduling Function. In Section IV we present an evaluation of 6TiSCH MSF performance on simple topologies with constant and varying traffic load. Finally, in Section V, we draw concluding remarks and suggest further work.

II. 6TiSCH

Industrial environments are prone to interference which limits the ability of a single-channel solution to provide reliable communication. Inspired by the existing WirelessHART and ISA100.11a standards [2], the IEEE Std 802.15.4-2015 [3] standard proposes a Medium Access Control (MAC) mechanism to improve the quality of communications for a wide range of applications, including industrial ones. This protocol combines channel-hopping with Time Division Multiple Access (TDMA) to achieve both high reliability against interference and very low energy consumption.

A. Time Slotted Channel Hopping (TSCH)

Under TSCH, transmissions are organized within a recurring *slotframe*, as presented in Fig. 1. In this slotframe, each individual transmission is scheduled as a pair of timeslot (horizontally in the time domain) and channel offset (vertically in the frequency domain). This atomic unit of transmission is called a *cell*. According to the standard, a slotframe contains 101 timeslots, each 10 ms long, and as many channel offsets as available physical radio channels, i.e., 16 in the 2.4GHz band. Each cell can be reserved for a specific node to receive and/or to transmit a packet. The cell can also be dedicated to a unicast link, or shared among multiple nodes, typically for control packets. In the latter case, the nodes use a contention-based method to access the channel.

978-1-7281-8086-1/20/$31.00

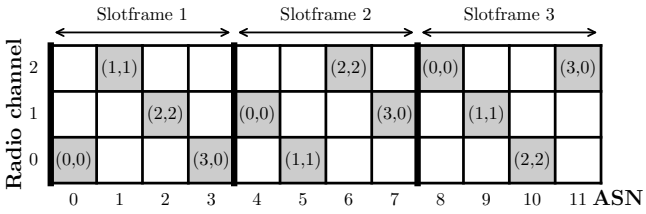

Figure 1. Recurring slotframe of size 4 with 3 radio channels. The same cells are scheduled in each slotframe and represented as (`timeslot`, `channelOffset`).

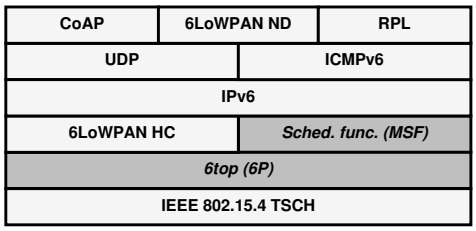

Figure 2. Network stack in the 6TiSCH architecture.

The channel offset does not directly map to the radio frequency. Instead, the actual frequency is determined using a hash function of the Absolute Sequence Number (ASN), an integer value that represents the time of the deployment, and the channel offset. It should be noted that in Fig. 1 the hash function maps the same scheduled cell, for instance (0,0), to a different physical radio channel on each occurrence of the slotframe. These two concepts, scheduling and channel-hopping, are at the core of TSCH. By spreading the communications over multiple channels, TSCH limits the impact of interference occurring on specific frequency bands, while the synchronous schedule-based approach avoids most collisions as most transmissions take place in contention-free cells.

Nodes wishing to join the network must synchronize themselves with the slotframe. Thus a special control frame at the MAC layer, known as the Enhanced Beacon (EB), is periodically sent over the air to announce the slotframe characteristics and beginning. This message, usually sent in the (0, 0) cell, contains, among other things, the size of the slotframe, the number of channels available and the current ASN value.

The IEEE Std 802.15.4-2015 TSCH standard does not define the strategies to construct and maintain the schedule of cells within the slotframes. Instead, the management of this schedule is left to an external entity. These solutions can be either centralizd, where a node is selected as a coordinator for the entire network, or distributed, where each node makes its decisions locally in collaboration with its neighbors. While the latter is more suited to larger unstable networks, the lack of a global network view makes it harder for these to ensure an efficient multi-hop communication.

Many distributed scheduling solutions have already been proposed in the literature [4]–[8]. For instance, Orchestra [4] uses hashes of the neighbors' MAC addresses to construct rendez-vous points in which to schedule transmissions.

Another solution currently worked on by the IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) [9] Working Group (WG) is the Minimum Scheduling Function (MSF) [10]. This solution provides a reactive algorithm which can quickly adapt to traffic variations and collisions, displacing conflicting cells or allocating new cells when needed. We present this scheduling function in more detail in Section III. The goal of this paper is to evaluate the performance of this under-standardization scheduling function.

B. 6TiSCH Architecture

The 6TiSCH WG envisions an IPv6-based wireless sensor network architecture [11] based on TSCH that aims for high-reliability packet delivery. To this end, it defines a network stack (Fig. 2) where IPv6 connectivity is achieved using well-known protocols such as the 6LoWPAN [12] shim layer with header compression (HC) and fragmentation, along with RPL [13] for routing and CoAP [14] as the application layer.

RPL organizes routing by constructing a *Destination Oriented Directed Acyclic Graph* (DODAG) that allows each node to reach the network root – usually the border router – through a *preferred parent*. The selection of preferred parents is based on the advertisement of *DODAG Information Object* (DIO) messages. Moreover, preferred parents act as clock sources to maintain the synchronization of the underlying TSCH timeslots.

In addition to these protocols, 6TiSCH also defines *scheduling functions* which implement distributed slotframe scheduling strategies and the 6TiSCH *operation sublayer* (6top) [15] which supports the negotiation of cells between neighboring nodes. Finally, it describes the minimal configuration required for nodes to join a 6TiSCH network [16].

C. 6TiSCH Operation Sublayer

The 6top layer is right above the link layer. The protocol part of this sublayer, called 6P [15], defines the messages and transaction mechanisms to *add*, *delete*, or *relocate* cells within the slotframe. Additionally, it also provides commands to *count*, *list*, or *clear* all the cells reserved for communication between two nodes as well as a signaling mechanism for proper operation of the scheduling functions. The decision of when to add or delete cells is left to a 6TiSCH Scheduling Function (SF).

Each 6top transaction consists of either 2 or 3 steps. In a 2-step transaction, the source node selects the candidate cells. In a 3-step transaction instead, it is the destination node which selects the candidate cells. The 6TiSCH MSF scheduling function presented in Section III only uses the 2-step transactions.

Fig. 3 illustrates an example of a 2-step ADD transaction. In this case, node A requests to node B the addition of two new cells to its own schedule. To this end, the scheduling function on node A proposes a list of three candidate cells and locks those in its schedule until a 6P response is received. When the request is successfully delivered to node B, indicated by the reception of a MAC-layer ACK by node A, node A also starts a timeout to abort the transaction if no response is received for its request. The scheduling function on node

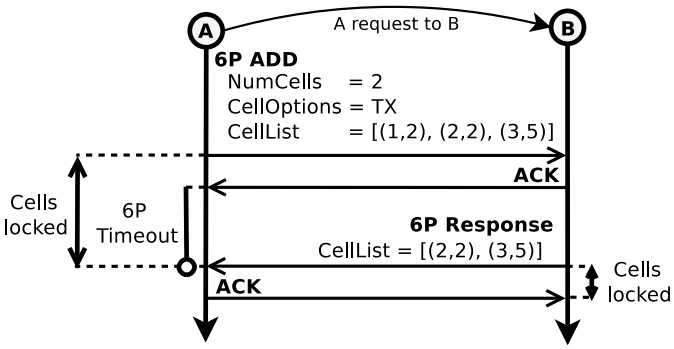

Figure 3. An example of a 2-step 6top ADD transaction. The request is made from node A to node B. Node A requests two cells among three proposed candidates. Node B responds with two selected cells among the candidates.

B selects two cells among the proposed candidates and locks those in its schedule until the response has been successfully received. Typically, 6P ADD transactions in 6TiSCH MSF request the negotiation of 1 cell among 5 candidates.

III. Minimal Scheduling Function

The 6P protocol only provides the necessary transactions to manipulate cells in each node's schedule. It is up to the scheduling functions to decide when to add or delete cells from those schedules. To this end, the 6TiSCH WG proposes a reactive and distributed scheduling solution known as the Minimal Scheduling Function (MSF) [10]. This scheduling function defines the bootstrapping process for a node to join the network and a subsequent mechanism for each node to adapt to traffic changes, routing changes, and schedule collisions.

A. Types of Cells

MSF relies on 3 different types of cells for its operations: the *minimal cell*, *autonomous cells* and *negotiated cells*. In case multiple cells are scheduled at the same slot and channel offset, the minimal cell has the highest priority, followed by autonomous cells.

The *minimal cell* is a single mandatory shared cell used to bootstrap the network [16] and ensure minimal connectivity. It is used to exchange the Enhanced Beacons advertising the network and its configuration, as well as routing information through the RPL DIO control packets. The minimal cell is usually located at timeslot 0 and channel offset 0.

MSF also makes use of a set of *autonomous cells* that act as default rendez-vous points to bootstrap unicast communications. Every node has a permanent Rx autonomous cell whose location in the slotframe is derived from a hash of its 64-bit Extended Unique Identifier (EUI64). On the other hand, Tx autonomous cells are allocated on-demand when no other unicast cell is available to send messages to a specific neighbor. In particular, they are used to transmit the initial messages to exchange keying material and negotiate via 6P the first cell to the preferred parent node. Sending through a Tx autonomous cell requires a contention-based method to access the channel, since the cell is shared by multiple neighbors.

Finally, MSF allocates *negotiated cells* that will be used by a node for communication and announcing itself to potential newcomers. Such cells are negotiated by a node with its neighbors through 6P transactions, according to the current traffic load.

B. Network Bootstrapping

A node expecting to join a 6TiSCH network must go through a series of steps before being able to transmit messages within the network. First, it must discover and synchronize with the network. Then, it must learn keying material and setup routing to its preferred parent. Finally, it must negotiate cells. This process can be divided into 6 steps detailed below.

1) **Channel selection:** Initially, the node expecting to join the network should choose a random radio channel to listen for an EB message advertising the network, which is sent from one of its neighbors. If the node does not hear any EB after some time, this may indicate that this specific radio channel is subject to interference. Thus the node should select another random radio channel and start again.
2) **Additional EBs:** Once the first EB has been received, the node should listen for additional EBs to discover its neighbors and to select its preferred neighbor as a Join Proxy (JP) to continue the join process. Once this JP has been selected, the minimal cell is configured on the joining node to enable communications.
3) **Join Process:** The node must now register to the network and learn the keying material. It does so by "talking" with a Join Registrar/Coordinator (JRC).
4) **Acquiring a RPL rank:** After the node has joined the network, it can receive the control messages, in particular RPL DIOs. Once at least one DIO is received the node can compute its own rank and select a preferred parent, as per [13].
5) **6P ADD to preferred parent:** Once the preferred parent has been selected, the node uses 6P to request from the parent one negotiated cell among 5 proposed candidates. This negotiated cell can be used only for unicast transmission to the preferred parent. This initial 6P request occurs over autonomous cells which are removed after transmission. Subsequent 6P requests will occur on any of the negotiated cells to the preferred parent.
6) **Send EBs and DIOs:** The node now starts sending DIOs and EBs through the minimal cell, allowing new devices to discover and join the network. To reduce contention in the minimal cell, the node should reduce the number of EBs and DIOs sent according to the number of neighbors.

C. Addition / Deletion Rules

MSF dynamically adapts the number of negotiated cells of each node. This happens in the three following cases. First, the available link-layer resources are adapted to the current traffic load. Second, a new preferred parent is selected, as part of RPL operations and cells must be re-negotiated. Finally, certain cells experiencing excessive schedule collisions need to be relocated.

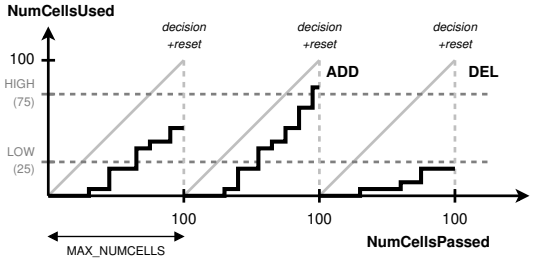

Figure 4. Decision to request/delete cells, with hysteresis.

1) Adapting to Traffic Changes: A node adapts its number of negotiated cells when it detects a significant increase or decrease in traffic. To this end, it estimates the traffic load over a recent window of time expressed as a number of cells. This is done by maintaining a pair of counters (*NumCellsUsed* and *NumCellsPassed*) per neighbor and per traffic direction. In the following discussion we only consider traffic going upstream, through the preferred parent. *NumCellsPassed* counts the elapsed number of negotiated cells to the preferred parent whether or not they resulted in a transmission, while *NumCellsUsed* counts the subset of those cells that *were used* for a transmission, whether or not that transmission was successful.

A node updates and adapts its schedule after a certain number of cells, *MAX_NUMCELLS*, has passed, that is, when *NumCellsPassed* $>$ *MAX_NUMCELLS*. At the time of decision, its estimate of the current traffic load is $\frac{NumCellsUsed}{MAX\_NUMCELLS}$ which is used *with hysteresis* to decide if cells must be requested or deleted. To this end, if *NumCellsUsed* $>$ *LIM_NUMCELLSUSED_HIGH*, then the node uses 6P to add a single negotiated cell. Otherwise, if *NumCellsUsed* $<$ *LIM_NUMCELLSUSED_LOW*, then the node uses 6P to remove a single negotiated cell. In any case, the node afterwards resets both counters (*NumCellsPassed*, *NumCellsUsed*) to 0. We illustrate this behavior in Fig. 4. The values used for *MAX_NUMCELLS*, *LIM_NUMCELLSUSED_HIGH* and *LIM_NUMCELLSUSED_LOW* are respectively, 100, 75 and 25, as recommended in [10].

2) Switching Preferred Parent: As part of the default operation of RPL, a node can switch its preferred parent when the link quality changes. When this occurs, the node should adjust its schedule accordingly. First, the node uses 6P to add the same amount of negotiated cells to its new preferred parent, as it had to the old preferred parent. Then, it issues a 6P clear to its old preferred parent to remove all previously negotiated cells. This operation is repeated for negotiated TX and RX cells.

3) Handling Schedule Collisions: Since the schedule is constructed in a distributed fashion, there is a possibility for two pairs of nearby neighbor nodes to schedule over the same cell (`timeSlot`, `channelOffset`). This could result in a collision if both pairs of nodes try to exchange packets at the same time.

A node detects such collisions with the use of two counters *per each negotiated TX cell*. *NumTx* counts the number of times a node tried to transmit a packet while *NumTxAck* counts the number of times such transmission is successful, that is, the number of transmissions for which an acknowledgment was received.

We define as the *Cell Delivery Ratio* (CDR) the ratio $\frac{NumTxAck}{NumTx}$ for cells where $NumTx > 0$. The value of both counters is divided by 2 when *NumTx* reaches 256. Thus, the counters can increment continuously without changing the value of the CDR.

The principle is that a cell subject to collisions would exhibit a CDR significantly lower than the other cells. Thus, to detect collisions, a node regularly issues the following sequence of actions: to ensure that the CDR value is statistically significant, the node waits until both counters were divided by 2 at least once before proceeding forward. When that has been done, it computes the CDR for each cell to its preferred parent and retains the maximum of those values. Then it relocates each cell whose CDR difference to the maximum is larger than a given threshold of *RELOCATE_PDRTHRES* with a default value of 50%.

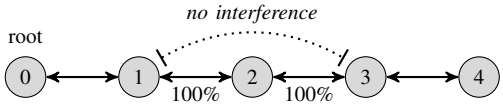

Figure 5. Linear topology with a link quality of 100% between adjacent nodes and no interference between non-adjacent ones.

IV. Evaluation

Two of the main features advanced by 6TiSCH MSF are the ability to adapt the network resources to the current traffic load of the network and to relocate these resources in case of collisions. In this section, we use the 6TiSCH simulator to provide an evaluation of MSF on two aspects. First we evaluate it with regular and constant traffic, then with varying traffic to assess the adaptation ability of MSF. We perform these evaluations on the simple linear topology presented in Section IV-A which allows us to investigate 6TiSCH MSF at a fundamental level.

A. Simulation Setup

To perform this evaluation, we use the 6TiSCH simulator [17]. This discrete-event simulator, written in Python, implements a careful abstraction of the 6TiSCH network stack. In particular it can accurately monitor the behavior of the Scheduling Function, the routing protocol, the impact of MAC layer drops for 6P transactions, and the response of the application. Note that this simulator does not reproduce a realistic PHY layer.

For most of our experiments we use a simple linear topology, illustrated in Figure 5, defined as a series of n nodes arranged linearly with a fixed link quality of 100% between each pair of adjacent nodes and 0% otherwise. The simulator implements the RPL Minimum Rank with Hysteresis Objective Function (MRHOF). However this does not impact parent selection as the nodes can only hear from their two immediate neighbors.

All our simulations use the default parameters presented in Table I. If a parameter changes for a particular experiment, it is stated explicitly in the text. The values provided are commonly found or recommended for 6TiSCH networks. The *EB/DIO* parameter is the probability for a node to send

Table I
DEFAULT PARAMETERS USED IN THE SIMULATIONS.

Parameter	Value
Timeslot duration	10 ms
Slotframe length	101 slots
EB/DIO probability	0.33
Packets size	90 Bytes
Retransmission	disabled

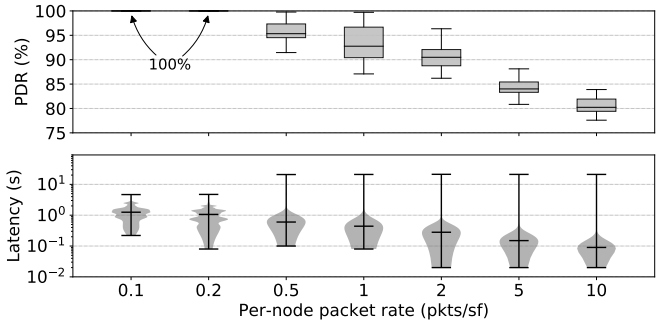

Figure 6. End-to-end PDR and latency for the application packets carried by node 2. Whiskers on the figures represent the minimum and maximum values, the middle horizontal line the median. In the PDR figure, the intermediate box lines represent the 25th and 75th percentiles.

an EB or DIO packet in the shared minimal cell. We also disabled retransmission at the TSCH level to force the use of available MAC layer resources instead of delaying packet losses as a retry into the TX queue. This allows us to observe the ability of MSF to send traffic on its own. Note that 6P requests are still retried by 6P itself as part of a 6top transaction.

Each simulation is repeated a large number of times for a fixed duration after all nodes have joined the network. To speed up the join process, we only start the more demanding application traffic after all nodes have joined the network. This is considered as $t=0$ s in our results. Finally, we also stop the application traffic 5 minutes before the end of the simulation to ensure that any packet in transit has time to reach its destination.

B. Constant Traffic

We evaluate the performance of MSF on a linear topology of 5 nodes as presented in Fig. 5. Each node from 1 to 4 generates a regular traffic with rate R ranging from 0.1 up to 10 packets/slotframe. Although the latter might seem excessive, we use it to simulate the load of a very large network. The simulation runs 50 times and for a duration of 30 minutes after all the nodes have joined the network. We focus on node 2 as it is the most susceptible to suffer from schedule collisions with the other nodes.

Fig. 6 shows the evolution of PDR and latency for node 2 as a function of traffic load. We observe that the PDR starts to drop with packet rates higher than 0.5 pkts/sf. On node 2, this corresponds to 1 pkt/sf of forwarded traffic from node 3 and 4 and 0.5 pkt/sf of locally generated traffic. The resulting 1.5 pkts/sf traffic overruns the single cell allocated to the parent and, thus,

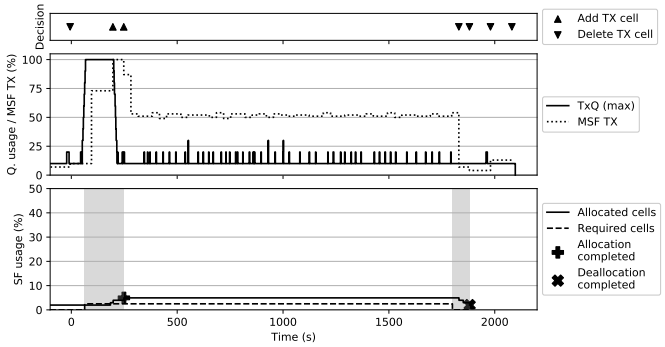

Figure 7. Evolution of allocated cells with time on node 2 with a generated traffic load of 0.5 pkts/sf/node. Allocation/deallocation periods are shaded in gray.

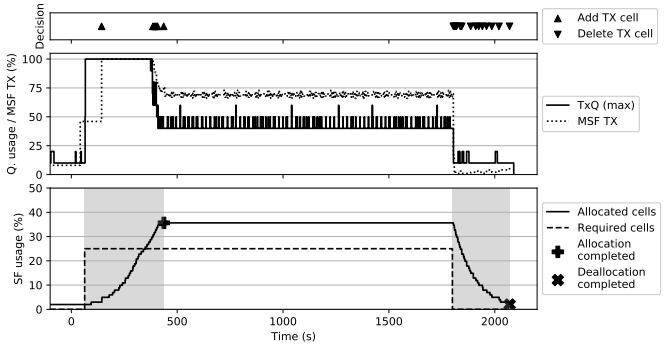

Figure 8. Evolution of allocated cells with time on node 2 with a generated traffic load of 5 pkts/sf/node. Allocation/deallocation periods are shaded in gray.

triggers the MSF traffic adaptation mechanism. When the queues become full and as long as the amount of required cells is not allocated, packets will be silently dropped, hence decreasing PDR.

Counter-intuitively, the latency for node 2 decreases with higher packet rates, except for outlier cases. This decrease in latency at higher traffic rates can be explained by the uniform distribution of more cells in the schedule. In that case, any packet has more opportunity to find a nearby cell to be sent on instead of waiting for the occurrence of a later slotframe. Also since the TX queue fills up as long as not enough resources are allocated for the traffic load, some packets can take a lot of time waiting in the queue to reach their destination.

Furthermore, we show the evolution of MSF traffic adaptation mechanism over time for a low traffic rate (Fig. 7) and a high traffic rate (Fig. 8). The MSF TX line shows the estimation of the negotiated cells usage over the last *MAX_NUMCELLS* window. Above 75%, MSF tries to add more cells, and below 25% to delete cells instead. The top part of the figure shows the decision by MSF to allocate new cells (up arrow) or to de-allocate existing cells (down arrow). The network starts with only one cell allocated which is not enough to send a traffic rate above 1 pkt/sf. We can see that as soon as the application starts sending packets, the transmission queue (TxQ) of node 2 immediately fills up to 100%. MSF TX quickly goes above 75% and MSF starts adding new cells to cope with the increased cells usage. This triggers an allocation period, shaded in grey, that lasts until

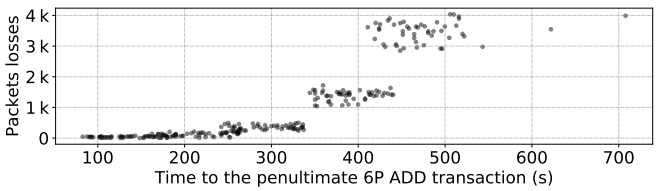

Figure 9. Correlation between the penultimate 6P ADD transaction and the number of losses observed for 50 runs and all packet rates.

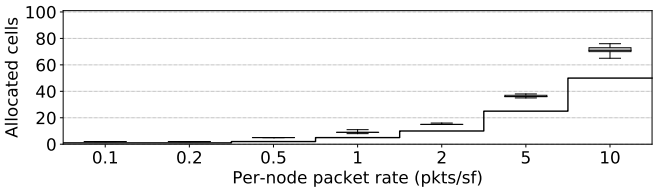

Figure 10. Number of cells allocated on node 2, as a function of per-node packet rate.

the cell usage decreases below 75%. This only happens once the resources are sufficient for the TX queue to not use all available cells. The same process happens in reverse when the application is stopped. The queue empties itself, and no cells are used for transmission. As a result, the cell usage goes below 25%, which triggers a deallocation period until no more cells can be deleted.

The vast majority of the losses happening in a 6TiSCH MSF network occur during these cell allocation period. When the TX queue is full, all new packets are dropped by node 2. It is only when resources are sufficient that the average empty rate of the queue equals its fill rate and packets are not dropped anymore. Thus, the length of those allocation periods directly impacts the PDR of the network. Fig. 9 illustrates this behavior with a correlation between the time to the penultimate 6P ADD transaction (X axis) and the number of packets losses on the network (Y axis). It shows that the number of observed losses increases with the time of the latest 6P ADD transactions, that is the time after which MSF has allocated the necessary resources for the current traffic load. Since MSF stops adding new cells as soon as MSF TX drops below 75%, variability in MSF TX can trigger an additional 6P ADD transaction long after the resources needed to avoid immediate packets losses have been allocated. For this reason, we measure the time to the penultimate instead of the last transaction.

We observe the rate of adaptation during these periods is not linear and increases with the number of allocated cells. Decisions to adapt are taken everytime *NumCellsPassed* $\geq$ *MAX_NUMCELLS*. As new cells are added, the time to reach *MAX_NUMCELLS* decreases, resulting in faster allocations. We also observe significant variation in the queue depth after MSF has converged. We hypothesize this is related to how uniformly cells are allocated within the slotframe. Clustered cells in the schedule increase the average distance between the cells, giving more opportunity for the queue to grow while waiting for a transmission cell.

We observed in Figs. 7 and 8 that the number of allocated cells was higher than required. In Fig. 10, we show for each traffic rate and 50 runs of the same configuration, the number of cells allocated by MSF on node 2, together with the theoretical number of cells required (line steps). The theoretical number of cells on node 2 is obtained as $N_{\text{th}} = \lceil 5 \times R \rceil$, where R is the per-node traffic rate. The factor 5 comes from the fact that node 2 receives data packets from node 3 and 4 and forwards them upstream along with its own packets. Let's consider the case of $R = 5$ pkts/sf. The theoretical number of cells required is 25 while the median (resp. maximum) number of allocated

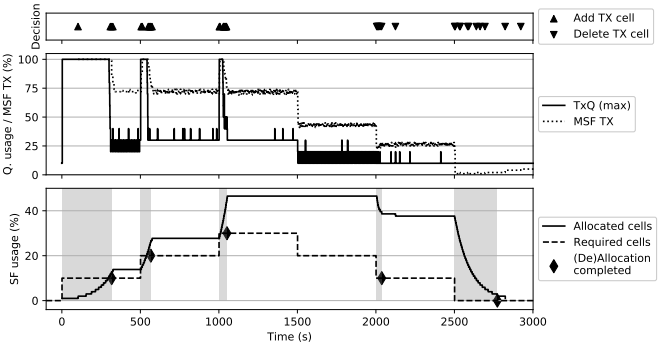

Figure 11. Evolution of allocated cells along time with a traffic load varying in the 0 – 30 pkt/sf range with rate change steps of 10 pkt/sf each. Allocation/deallocation periods are shaded in gray.

cells in our experiments is 36 (resp. 38). Note that over-provisioning was expected because of the cell usage threshold ($\frac{\textit{LIM\_NUMCELLSUSED\_HIGH}}{\textit{MAX\_NUMCELLS}} = 75\%$) value used for deciding for the addition of new cells. The amount of observed over-provisioning can be estimated as $N_{\text{obs.}} = \frac{\textit{MAX\_NUMCELLS}}{\textit{LIM\_NUMCELLSUSED\_HIGH}} \times N_{\text{theo.}}$. For $R = 5$, that gives $N_{\text{obs.}} \approx 33$, which is close to the observed results.

C. Changing Traffic

This section focuses on MSF ability to quickly allocate or deallocate resources when the traffic load changes. To do so, we use a simpler setup with only two nodes: the root and one leaf node sending traffic at a packet rate that periodically changes. Every 500 seconds, the sending application cycles through the following rates: 10, 20, 30, 20, 10 and finally back to 0 pkts/sf. We measure the time required from the moment the packet rate changed to the moment we reach a stable schedule in the slotframe.

Fig. 11 shows the evolution of several parameters along time for a single run of this simulation. Similar to Figs. 7 and 8, the figure is split into three parts. The middle one shows the evolution of the transmit queue (TxQ) and the MSF estimation of the traffic load (MSF TX). The bottom part shows the evolution of the number of allocated cells along with the theoretical minimum number of cells. The top part shows when MSF decides to allocate new cells (up arrow) or to de-allocate existing cells (down arrow).

The sending application starts at $t = 0$. The traffic rate suddenly goes from 0 to 10 pkt/sf and as a consequence, TxQ jumps to 100% occupancy as there are insufficient cells. MSF kicks in and slowly allocates new cells through 6P ADD requests. We can notice that the rate at which new cells are allocated rapidly increases as it takes less and less time for

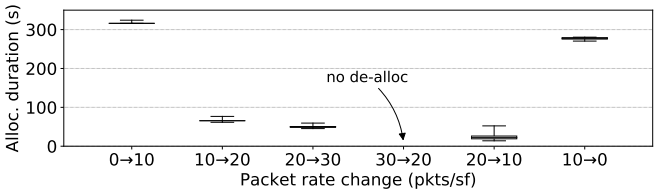

Figure 12. Duration for MSF to reach a stable state and allocate all necessary cells after a change in traffic rate.

MAX_NUMCELLS to pass. At $t=316$ s, MSF has converged to a stable state; the slotframe now contains enough cells to carry the traffic load. At $t=500$ s, the traffic rate jumps from 10 to 20 pkts/sf, leading to another round of cell allocations that ends at $t=565$ s. Although this jump in traffic rate is equal in intensity to the first one, the time to adapt was much shorter. At $t=1000$ s, the last increase in traffic rate takes place, jumping from 20 to 30 pkts/sf. It requires an even shorter convergence time (50 s).

After $t=1500$ s, the traffic decreases from 30 to 20 pkts/sf. However MSF withholds the decision to de-allocate cells as MSF TX does not drop below the 25% limit. This results in a higher over-provisioning level compared to what was observed with a constant traffic load in Section IV-B. At $t=2000$ s, the traffic decreases again from 20 to 10 pkts/sf. This time, MSF triggers de-allocations but only for a handful of cells until it reaches the lower limit of 25%. After $t=2500$ s, the traffic drops back to 0 pkt/sf resulting in a MSF TX of $\approx 0\%$. Hence, MSF de-allocates all but one cell for a duration of 279 s.

Fig. 12 shows the time required to allocate or de-allocate cells after each change of traffic rate for 100 runs of the same configuration. Those durations show little variability and match the single run presented in Fig. 11. With jumps in traffic rate of equal intensity, the duration to reach a stable state varies a lot depending on the amount of cells already present in the slotframe, with longer durations for lower the slotframe usage.

V. Conclusions

The deployment of Industrial IoT networks requires that they can quickly adapt to traffic changes in interference-prone environments. The MSF provides a distributed scheduling function on top of IEEE Std 802.15.4-2015 TSCH to adapt MAC layer resources to the requirements of the network along with the relocation of those resources in case of collisions. We employed the 6TiSCH simulator to evaluate the ability of MSF to allocate the network resources. We observed that without retransmissions, packet losses can appear as soon as the traffic adaptation mechanism becomes necessary. This is to be expected considering the reactive nature of MSF.

The duration to allocate those necessary resources has a direct impact on the amount of losses seen during traffic load changes. We have seen that the rate at which those resources are allocated can change considerably and depends on the number of cells already allocated in the slotframe. We also observed that MSF is subject to over-provisioning of the network resources and frequently allocates or keeps more cells than are required to send the current traffic load. This is even more pronounced in the case of a varying traffic load where MSF would reluctantly give up cells that it previously allocated. As future work, we plan to study MSF modifications to accelerate the convergence and reduce losses when traffic changes occur and reduce the amount of over-provisioning, especially in the case of varying traffic load.

Acknowledgments

We thank Maximilien Charlier, Jérémy Dubrulle and Jeremy Gheysen for helping us to deepen our understanding of TSCH, RPL and 6TiSCH MSF. This work was supported by the European Regional Development Fund through the IDEES project portfolio.

References

[1] S. Duquennoy, J. Eriksson, and T. Voigt, "Five-nines reliable downward routing in RPL," *CoRR*, vol. abs/1710.02324, 2017. [Online]. Available: http://arxiv.org/abs/1710.02324

[2] S. Duquennoy, A. Elsts, A. Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *Proc. IEEE DCOSS 2015*, 2017.

[3] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, Apr. 2016.

[4] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. ACM SenSys 2015*. ACM, 2015, pp. 337–350.

[5] A. Aijaz and U. Raza, "Deamon: a decentralized adaptive multi-hop scheduling protocol for 6TiSCH wireless networks," *IEEE Sensors Journal*, vol. 17, no. 20, pp. 6825–6836, 2017.

[6] R.-H. Hwang, C.-C. Wang, and W.-B. Wang, "A distributed scheduling algorithm for IEEE 802.15. 4e wireless sensor networks," *Computer Standards & Interfaces*, vol. 52, pp. 63–70, 2017.

[7] F. Theoleyre and G. Z. Papadopoulos, "Experimental validation of a distributed self-configured 6TiSCH with traffic isolation in low power lossy networks," in *Proc. ACM MSWiM 2016*. ACM, 2016, pp. 102–110.

[8] T. P. Duy, T. Dinh, and Y. Kim, "Distributed cell selection for scheduling function in 6TiSCH networks," *Computer Standards & Interfaces*, vol. 53, pp. 80–88, 2017.

[9] P. Thubert, T. Watteyne, M. R. Palattella, X. Vilajosana, and Q. Wang, "IETF 6TSCH: Combining IPv6 connectivity with industrial performance," in *Proc. IMIS 2013*. IEEE, 2013, pp. 541–546.

[10] T. Chang, M. Vuini, X. Vilajosana, S. Duquennoy, and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," Internet Engineering Task Force, Internet-Draft draft-chang-6tisch-msf-10, Dec. 2019, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6tisch-msf-10

[11] P. Thubert, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4," Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-architecture-28, Oct. 2019, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6tisch-architecture-28

[12] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," RFC 4944 (Draft Standard), Internet Engineering Task Force, 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[13] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL : IPv6 routing protocol for low power and lossy networks," RFC 6550 (Draft Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[14] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[15] T. W. Q. Wang, X. Vilajosana, "6TiSCH Operation Sublayer (6top) Protocol (6P)," RFC 8480 (Draft Standard), Tech. Rep. 8480, Nov. 2018. [Online]. Available: http://www.ietf.org/rfc/rfc8480.txt

[16] X. Vilajosana, K. Pister, and T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," RFC 8180, May 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8180.txt

[17] E. Municio, G. Daneels, M. Vučinić, S. Latré, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6TiSCH networks," *Transactions on Emerging Telecommunications Technologies*, 2018.