Journal of Signal Processing Systems
https://doi.org/10.1007/s11265-021-01668-w

Thorough Performance Evaluation & Analysis of the 6TiSCH Minimal Scheduling Function (MSF)

David Hauweele[1] · Remous-Aris Koutsiamanis[2,3] · Bruno Quoitin[1] · Georgios Z. Papadopoulos[4]

Received: 18 December 2020 / Revised: 11 April 2021 / Accepted: 22 April 2021

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 prone to over-provisioning of the network resources, especially in the case of varying traffic load. We propose a mathematical model to predict the convergence pattern of MSF. Finally we investigate the impact of varying parameters on the behavior of the scheduling function.

Keywords Internet of things · IoT · Industrial IoT · IIoT · IEEE Std 802.15.4-2015 · TSCH · 6TiSCH · 6top · 6P · MSF

1 Introduction

The Industrial Internet of Things (IIoT) consists of a large collection of wireless sensors and actuators for various industrial applications. Sensor nodes periodically transmit their measurements to a controller which, based on this continuous feedback process, may trigger a reaction by enabling the actuators.

✉ David Hauweele
david.hauweele@umons.ac.be

Remous-Aris Koutsiamanis
remous-aris.koutsiamanis@imt-atlantique.fr

Bruno Quoitin
bruno.quoitin@umons.ac.be

Georgios Z. Papadopoulos
georgios.papadopoulos@imt-atlantique.fr

[1] University of Mons (UMONS), Mons, Belgium
[2] IMT Atlantique / DAPI, STACK (LS2N/Inria), IRISA, France
[3] IMT Atlantique, Inria and CNRS(LS2N), Nantes, France
[4] IMT Atlantique, IRISA, Nantes, France

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. Moreover, it comes with strict requirements of very high reliability, low latency, and low jitter on data transmission. Toward that goal, Industry 4.0 is expected to rely heavily on the IIoT by deploying Internet of Things (IoT) technologies for connecting management, reporting, sensing, and control interfaces purposes. 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.

Recently, wireless technologies have been used with good results in terms of reliability [12] and latency [20]. However, because of strict constraints on available network resources and required energy efficiency, assumptions are made about the characteristics of the served traffic, such as constant rate. Moreover, replacing legacy, wire-based infrastructure requires the ability to quickly adapt to changing traffic.

To compensate for these shortcomings and to allow industrial use of these devices, a set of protocols have been designed and developed in the standardization community to enable industrial networks. Recently, the Institute of Electrical and Electronics Engineers (IEEE) has defined a deterministic Medium Access Control (MAC) protocol named IEEE Std 802.15.4 Time Slotted Channel Hopping (TSCH). It relies on a strict schedule of non-interfering transmissions to mitigate the potential collisions, and on a slow channel-hopping strategy to combat external interference. Typically, a schedule is based on a matrix that is composed of *cells*, defined by a timeslot (the transmission time) and a channel offset (the radio channel). Thus, the scheduling algorithm is in charge of allocating a set of transmission and/or reception cells to each potential transmitter and receiver, respectively. To this end, the Internet Engineering Task Force (IETF) has introduced a still work-in-progress functionality known as the 6TiSCH *Minimal Scheduling Function* (MSF) [5], which allows the negotiation and reservation of network resources in an on-demand manner. In this paper, we illustrate shortcomings that its use brings to the IIoT use case, namely overprovisioning and slow convergence time.

In this paper, we first introduce 6TiSCH MSF in detail. We then assess its performance by studying the behavior and reactivity of MSF with varying traffic load. This article extends [15], making the following additional contributions:

- It presents a thorough literature review on scheduling functions that are proposed so far.
- It comes with additional performance evaluation results that further demonstrate the issues of MSF.
- It proposes a mathematical model of the convergence time.
- Furthermore, it investigates the impact of varying MSF parameters such as the number of slots used to estimate traffic load (*MAX_NUMCELLS*) and LOW/HIGH decision thresholds.

The rest of this paper is organized as follows. Section 2 presents a description of 6TiSCH. Then, in Section 3, we provide a detailed background on the Minimal Scheduling Function and we discuss the related work in Section 4. In Section 5, we present our exhaustive performance evaluation campaign of 6TiSCH MSF over simplified topologies with constant and varying traffic patterns. In Section 6 we vary three MSF parameters and discuss their impact on convergence time and overprovisioning. Finally, in Section 7, we draw concluding remarks and suggest potential further work.

2 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 [11], the IEEE Std 802.15.4-2015 [1] 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.

2.1 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, e.g., 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.

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

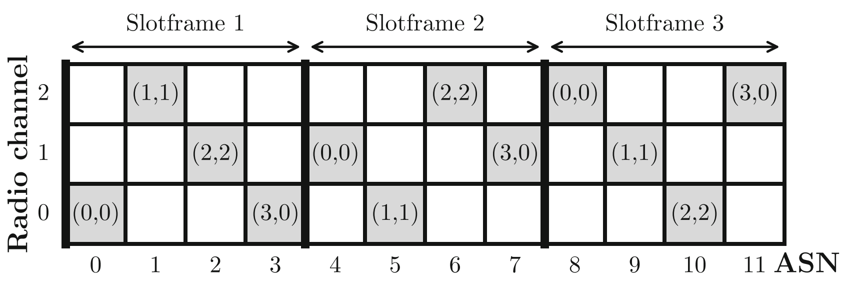


Figure 1 Recurring slotframe of size 4 with 3 radio channels. The same cells are scheduled in each slotframe and represented as (`timeslot`, `channelOffset`).

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 centralized, 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 and unstable networks, the lack of a global network view makes it harder for these to ensure efficient multi-hop communication.

Another solution, the Minimal Scheduling Function (MSF) [5], currently worked on by the IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) [30] Working Group (WG), provides a reactive algorithm which can

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 3. The goal of this paper is to evaluate the performance of this under-standardization scheduling function.

2.2 6TiSCH Architecture

The 6TiSCH WG envisions an IPv6-based wireless sensor network architecture [29, 33] 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 [21] shim layer with header compression (HC) and fragmentation, along with RPL [34] for routing and CoAP [28] 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) [25] which supports the negotiation of cells between neighboring nodes. Finally, it describes the minimal configuration required for nodes to join a 6TiSCH network [32].

2.3 6TiSCH Operation Sublayer

The 6top layer is right above the link layer. The protocol part of this sublayer, called 6P [25], 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 and how many cells to add or delete 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 3 only uses the 2-step transactions.

Figure 3 illustrates an example of a 2-step ADD transaction. In this case, node A requests from node B the addition of two new cells to its own (A's) 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

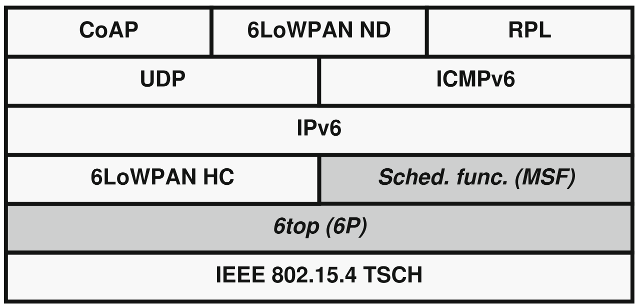


Figure 2 Network stack in the 6TiSCH architecture.

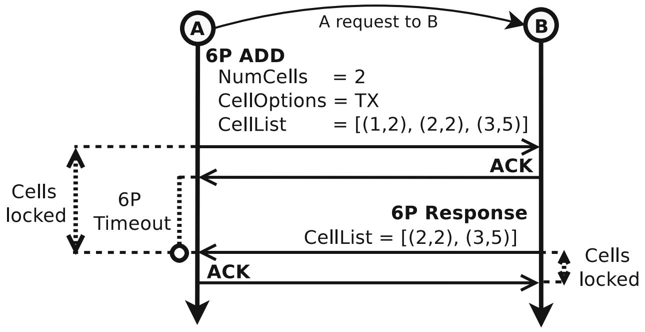


Figure 3 An example of a 2-step 6top ADD transaction. The request is made from node A to node B. Node A requests two cells among three proposed candidates. Node B responds with two selected cells among the candidates.

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 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.

3 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) [5]. 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.

3.1 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 [32] 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 (reception) 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 (transmission) 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.

3.2 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 EBs 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 [34].
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.

3.3 Addition / Deletion Rules

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

3.3.1 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 scheduled 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 [5].

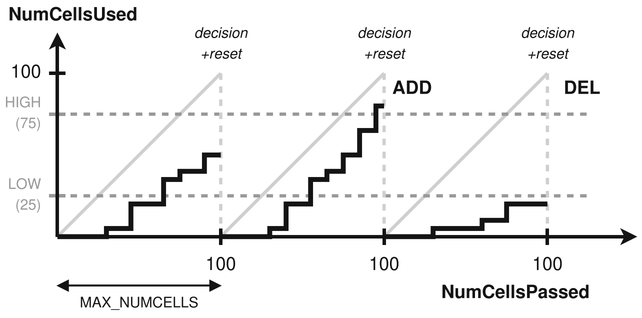


Figure 4 Decision to request/delete cells, with hysteresis.

3.3.2 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.

3.3.3 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 *LRELOCATE_PDRTHRES*, with a default value of 50%.

4 Related Work

Scheduling IEEE 802.15.4 communications has received a lot of attention from the computer networking community since the release of amendement 802.15.4e [13] in 2012. Even though the revision of the standard introduces a new access mode named TSCH (*Time-Slotted Channel Hopping*), it does not specify how the TSCH slotframe must be established.

Scheduling approaches can be divided into centralized and distributed approaches. With centralized scheduling approaches, a controller typically has *a priori* knowledge of the network topology, interfering links and traffic demand and is able to compute a close to optimum schedule. Many different underlying optimisation objectives are possible, but a typical concern is to minimize the slotframe length while being able to serve the whole traffic load. The incentive being that short slotframes imply low latency, an important requirement in many industrial applications. An example of an early centralized scheduling approach is TASA [23], which targets convergecast applications. To obtain compact slotframes, TASA favors simultaneous non-conflicting concurrent communications on different channels. It relies on a graph coloring heuristic to achieve this goal. Most centralized approaches leave the problem of collecting the topology, interferences and demand statistics as further work. Moreover, in case of dynamic networks where the topology and demand can vary through time, even though recomputing a new schedule is possible, updating the network with the new slotframes while it is running often remains an open question.

In distributed approaches, scheduling occurs on the nodes themselves, based on a partial knowledge of the topology typically limited to their immediate neighborhood. Moreover, the estimation of the traffic demand is based on measurements only. Interfering links cannot be known *a priori* but can be detected when experiencing failed transmissions. As such, distributed scheduling seems to be harder to solve than centralized scheduling. However, it comes with the benefit of less signaling overhead since there is no need to maintain an up-to-date view of the whole network topology and scheduling decisions only affect the local slotframes. DeTaS [2] is an example decentralized algorithm for convergecast applications that produces a schedule in a hierarchical manner, over a pre-existing spanning tree. Nodes start by requesting bandwidth allocation from their parent. Those requests are aggregated at each level of the tree until they reach the root/sink. Allocation of slotframe cells is then performed in the reverse, top-down direction. Orchestra [10] relies on the RPL routing protocol to decide each node's parent. RPL signaling is performed in a shared slot common to all nodes. The parent of a node is used as a next-hop to reach the sink and as a clock source to maintain TSCH temporal synchronization. In addition to this, Orchestra introduces MAC address hash-based rendez-vous cells between direct neighbors (an approach used earlier with IEEE 802.11 [3]). Alice [18] is a variant of Orchestra that uses link-based hashing for rendez-vous cells.

We refer the reader to the survey published by Hermeto et al. [16] for a more extensive coverage of the literature and now focus on related work specific to scheduling within the 6TiSCH framework (see Section 2). This framework supports negotiation of cells between neighbors through the 6P protocol, leaving the questions of when and what cells to allocate to a *scheduling function*. The Minimal Scheduling Function (MSF) [5] proposed by the IETF is still at the draft stage at the time of this writing but is undergoing last reviews. Other scheduling functions have been proposed in the literature. Distributed PID-based scheduling [8] explores how to adjust the amount of allocated cells to the traffic load through a control process implemented as a *Proportional Integral Derivative* (PID) feedback loop. The On-the-Fly scheduling function is another scheduling function that once was a candidate for IETF standardization [9, 24, 26]. OTF schedules cell allocations based on an estimation of the current traffic load. It does not document however how this estimation is performed. More recently, a small study of the *MAX_NUMCELLS* parameter of MSF was carried out leading to the A-MSF variant [4]. A performance evaluation of different scheduling functions was performed by Righetti et al. [27] through simulation and real-world experiments. Their evaluation also considers the interaction between TSCH scheduling and the dynamics of routing, as well as the impact of 6P transaction failures that can significantly delay the (de-)allocation of cells. Based on their analysis, they propose a revised version of the OTF scheduling function, named E-OTF, that takes into account the queue occupancy and the expected retransmission count (ETX). When the queue depth is above some threshold, additional cells are allocated to quickly drain the queue. The ETX corrects the number of required cells to account for packet losses and the corresponding retransmissions.

The Low Latency Scheduling Function (LLSF) [6] exploits the daisy-chain technique, while it tackles the potential unreliable radio links by adding over-provisioning cells. In order to minimize the buffering delay, whenever a node allocates a transmission cell in its schedule, it also allocates a receiving cell as close as possible in the slotframe. However, when there is a change in the link quality, the whole schedule along the path may require changing.

Daneels et al. propose the Recurrent Scheduling Function (ReSF) [7], which aims to achieve low end-to-end latency for wireless sensor multi-hop networks. ReSF considers that each node is aware of its packet generation period, and it reserves a series of timeslots (reception and transmission cells) back-to-back along the path from source to sink. The reserved timeslots are activated only when the traffic is expected for energy saving purposes. ReSF addresses the potential packet losses by including additional timeslots depending on the radio link quality. However, ReSF may introduce scheduling collisions, which occur when two or more reservations on a particular node employ the same cell at the same time. Moreover, ReSF does not guarantee bounded delay, because if the over-provisioning timeslots are insufficient to successfully transmit a data packet, it will remain in the queue for the next slotframe.

In [19], the authors present the Low-latency Distributed Scheduling Function (LDSF) that relies on the organization of the slotframe in smaller sub-slotframes, called blocks. Each transmitting node selects the corresponding set of blocks, depending on its hop distance from the root node, so that retransmission opportunities are automatically scheduled. Furthermore, for energy saving purposes, each node can turn off its radio once its data packet is successfully transmitted, i.e., after an acknowledgement is received. However, LDSF comes with strong assumptions. For instance, it considers that each node has no more than 5 children which makes it a non-scalable scheduling function. Moreover, it considers that all nodes have exactly the same traffic type, i.e., Constant Bit Rate (CBR).

Hamza and Kaddoum [14] proposed the Enhanced Minimal Scheduling Function. The core of their proposal is to estimate the average traffic load of a node during a window of a few past slotframes. The traffic load is then modelled as a Poisson process with that estimate as the mean and used to then predict the required number of cells for future slotframes. If the currently allocated number of cells is lower than the worst-case prediction, additional allocations are triggered. EMSF was evaluated using the OpenWSN simulator, leading to improved results in terms of latency, queue depth and number of 6P transactions. The authors do not evaluate to what extent EMSF causes over-provisioning.

In [17], the authors present the On-Demand TSCH Scheduling with Traffic-Awareness (OST) scheduling technique, which allocates per-link cells on two principles: it allocates some cells (called periodic) on a long-term basis based on average traffic between the link nodes and it also allocates additional temporary cells on-demand for bursty traffic that exceeds the current periodic traffic allocation. This work uses existing data and ACK packets to exchange scheduling information to reduce overhead instead of the standardised 6P protocol. This, however, means that this is not interoperable with other schedulers that do use 6P and additionally the available payload space in all data packets is reduced. The method for on-demand allocation piggy-backs the information about where to schedule the additional cells on data packets depending on the length of the sender's queue. The successful reception of a packet will allocate the needed resources for the subsequent packets in the sender's queue. However, this technique will suffer if the link is unreliable since the inability to schedule more temporary cells will increase packet losses and further reduce reliability.

Recently, Amezcua Valdovinos et al. [31] proposed the Channel Ranking Scheduling Function (CRSF), another 6TiSCH scheduling function that estimates the number of required cells as the current queue depth minus the number of allocated cells and then pass it through a Kalman filter. In addition to this, a channel selection mechanism is introduced that relies on a composite metric including RSSI, PDR and background noise, all measured passively. A drawback of CRSF observed by the authors is that it strongly increases the packet latency compared to MSF in certain scenarios such as when experiencing bursty traffic.

All the scheduling functions discussed so far focus on performance metrics such as the PDR and latency but do not consider the time of convergence and overprovisioning as we do in this paper.

5 Evaluation

Two of the main features advanced by 6TiSCH MSF are the ability to adapt the allocated 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 5.1. This linear topology with "perfect" link qualities allows us to investigate 6TiSCH MSF at a fundamental level. We focus on the allocation of cells, and the problem we discuss occurs without the need of interference. In fact, using more complicated and realistic topologies would only exacerbate the problem and hinder the study of its root causes.

5.1 Simulation Setup

To perform this evaluation, we use the 6TiSCH simulator [22].

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 the purpose of our analysis, we also modified the 6TiSCH simulator to extract additional information during the simulations.

For most of our experiments we use a simple linear topology, illustrated in Fig. 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 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 1. 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 an EB or DIO packet in the shared minimal cell. We also disabled retransmissions 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.

5.2 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 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.

Figure 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

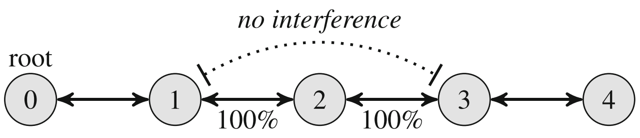


Figure 5 Linear topology with a link quality of 100% between adjacent nodes and no interference between non-adjacent ones.

Table 1 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
TX queue size	10 pkts

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, 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, a packet waiting to be sent has more opportunities to find a nearby cell to be sent on instead of waiting for the occurrence of the next slotframe. Additionally, since the TX queue fills up as long as insufficient resources are allocated for the traffic load, some packets can take a lot of time waiting in the queue to reach their destination.

Figure 7 further details the evolution of latency for node 2 and distinguish the packets sent during the allocation of resources (Fig. 7a) and after all the necessary resources have been allocated (Fig. 7b). We do not show the traffic rates 0.1 and 0.2 pkts/sf that are too low to trigger the MSF traffic adaptation mechanism. Accumulation of packets during the resource allocation period fills the TX queue, delaying packet transmission. As a result during this period, very high latencies occur on all traffic rates. Latencies measured for packets sent after all resources have been allocated do not show the same extreme values.

Furthermore, we show the evolution of the MSF traffic adaptation mechanism over time for a low traffic rate (Fig. 8) and a high traffic rate (Fig. 9). In the middle part of the figure, the MSF TX line shows the estimation of the negotiated cells usage over the last *MAX_NUMCELLS* window, while MSF RX does the same for negotiated RX cells. 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 deallocate existing cells (down arrow). The bottom part shows the current allocation of RX and TX cells. The dashed line indicates the minimum number of cells required to carry the traffic rate.

The network starts with only one cell allocated which is not enough to send a traffic rate above 1 pkt/sf. We can see

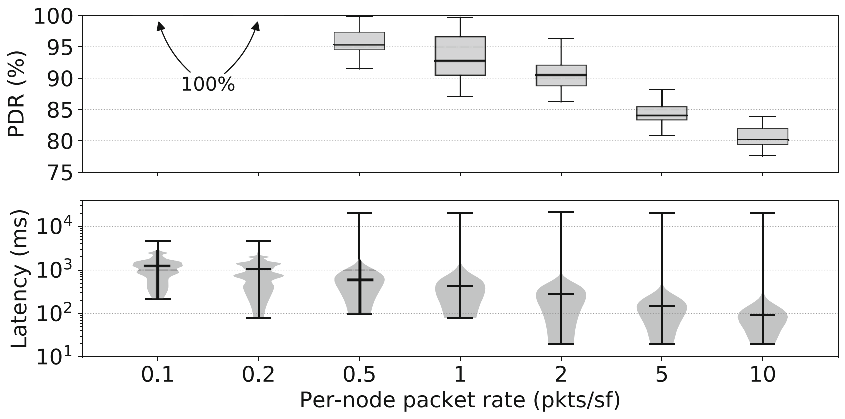
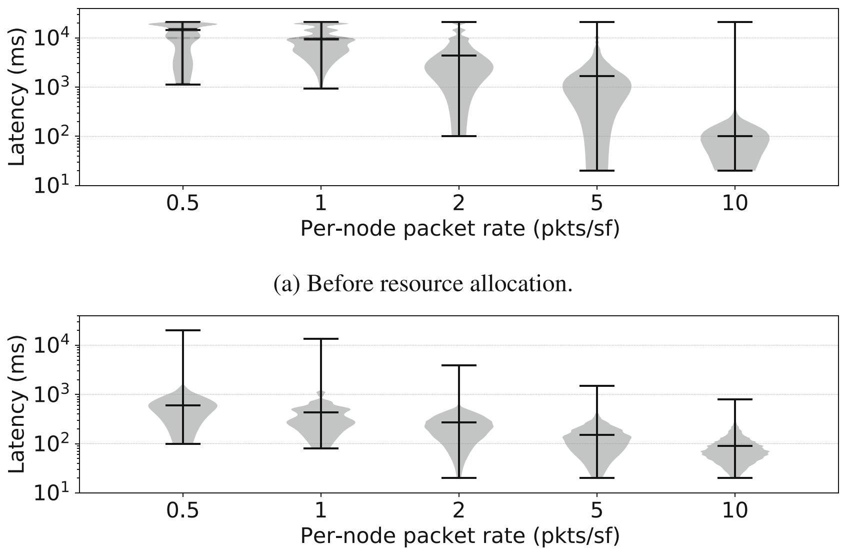


Figure 6 End-to-end PDR and latency for the application packets carried by node 2. Whiskers on the figures represent the minimum and maximum values, the middle horizontal line the median. In the PDR figure, the intermediate box lines represent the 25th and 75th percentiles.



(a) Before resource allocation.

(b) After resource allocation.

Figure 7 Latency for the application packets carried by node 2 before and after allocating the necessary resources. Whiskers on the figures represent the minimum and maximum values, the middle horizontal line the median.

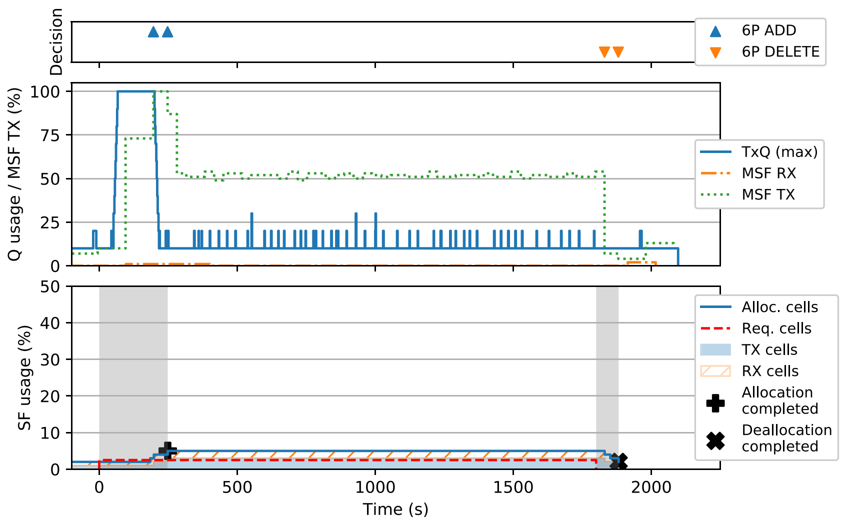


Figure 8 Evolution of allocated cells with time on node 2 with a generated traffic load of 0.5 pkts/sf/node. Top: cell allocation (up arrow) and deallocation (down arrow). Middle: MSF TX and RX estimators and transmit queue depth. Bottom: current allocation of RX and TX cells, theoretical number of required cells and allocation/deallocation periods (shaded in gray).

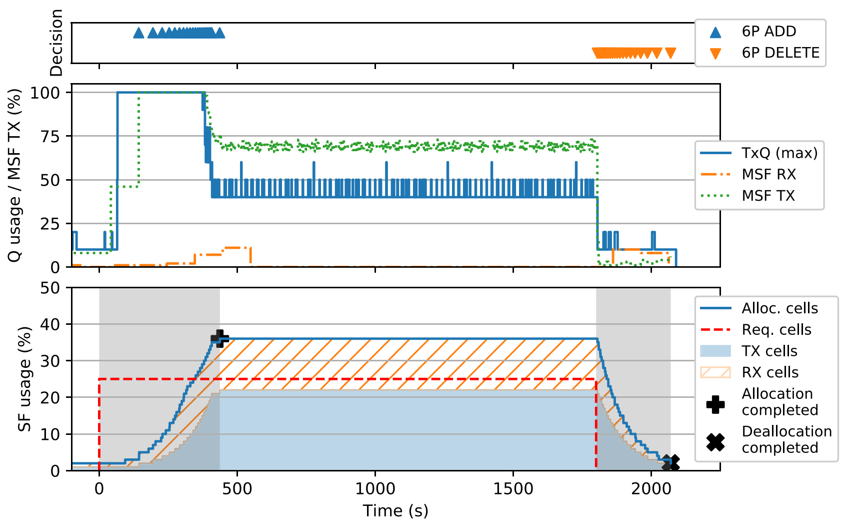


Figure 9 Evolution of allocated cells with time on node 2 with a generated traffic load of 5 pkts/sf/node. Allocation/deallocation periods are shaded in gray.

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 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, cell usage goes below 25%, which triggers a deallocation period until no more cells can be deleted.

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 whenever *NumCellsPassed* $\geq$ *MAX_NUMCELLS*. As new cells are added, the time to reach *MAX_NUMCELLS* decreases, resulting in faster allocations. We also observe a 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. 8 and 9 that the number of allocated cells was higher than required. In Fig. 10, we show for each traffic rate and 50 randomized 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 $N_{\text{req}} = \lceil 5 \times R \rceil$, 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 10 RX and 15 TX cells, for a total of $N_{\text{req}} = 25$ cells, while the median (resp. maximum) number of allocated cells in our experiments is 36 (resp. 38). Note that over-provisioning was expected because additional cell allocation will stop only when the MSF TX estimator falls below the high cell usage threshold, that is when MSF TX $\leq \frac{LIM\_NUMCELLSUSED\_HIGH}{MAX\_NUMCELLS} = 75\%$. The theoretical number of cells, taking into account over-provisioning, can be estimated by Equation 1. For $R = 5$, that gives $N_{\text{req}} Ovp \approx 33$, which is close to the observed results.

$$N_{\text{ovp}} = \frac{MAX\_NUMCELLS}{LIM\_NUMCELLSUSED\_HIGH} \times N_{\text{req}} \qquad (1)$$

5.3 Changing Traffic

This section focuses on MSF's ability to 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.

Figure 11 shows the evolution of several parameters along time for a single run of this simulation. Similar to Figs. 8 and 9, the figure is split into three parts. The middle one shows the evolution of the transmit queue length (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 deallocate 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 is activated and slowly allocates new cells through 6P ADD requests. We can notice that the rate at

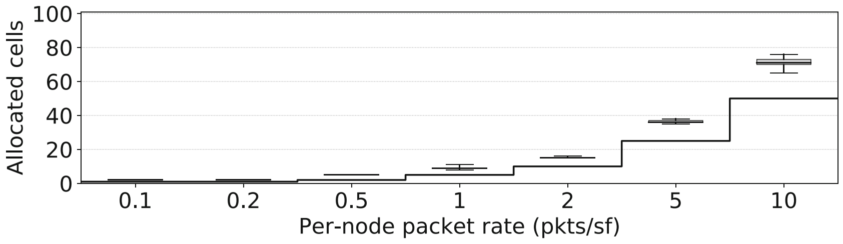


Figure 10 Number of cells allocated on node 2, as a function of per-node packet rate. The boxes represent the amount of cells allocated across multiple runs of the same simulation. The black line represents the minimum amount of cells, N_{req}, required to transport the CBR traffic.

which new cells are allocated rapidly increases as it takes less and less time for a period of *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 deallocate 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 5.2. At $t = 2000$ s, the traffic decreases again from 20 to 10 pkts/sf. This time, MSF triggers deallocations 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 value of MSF TX of $\approx 0\%$. Hence, MSF deallocates all but one cell during a period of 279 s.

Figure 12 shows the time required to allocate or deallocate cells after each change of traffic rate for 100 randomized 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 observed with lower slotframe usage.

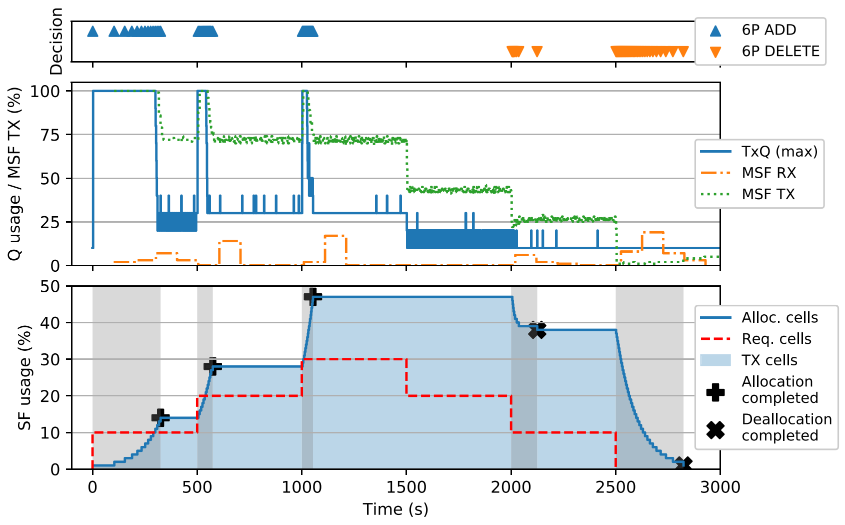
5.4 Convergence Model

In this section, we provide a model of the time required by MSF to converge to a new allocation state. Let's first consider the simple case where the number of TX cells is increased from k cells to $k + 1$ cells. To estimate the current traffic load, MSF looks at the fraction of allocated cells that are currently used. It does so during an *estimation round* that lasts for *MAX_NUMCELLS* allocated cells. During this round MSF counts the number of used cells (*NumCellsUsed*) and the number of allocated cells (*NumCellsPassed*). It then estimates the load by looking at the ratio of these counters.

$$\frac{NumCellsUsed}{NumCellsPassed} > 75\% \Rightarrow \text{allocate new cell}$$

To help understand this process, we rely on the example depicted in Fig. 13 where a node requests an additional TX cell to its parent. To simplify the discussion, the slotframe

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.



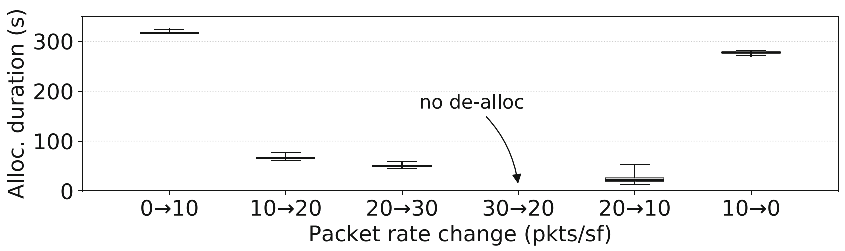


Figure 12 Duration for MSF to reach a stable state and allocate all necessary cells after a change in traffic rate.

length is 7 and the value of *MAX_NUMCELLS* is 8. Initially, the node slotframe contains 2 TX cells (TSN 1 and 4) and 1 RX cell (TSN 2). An estimation round extends from the 5th slot of the first slotframe to the 2nd slot of the 5th slotframe. The round lasts 26 slots or approximately 4 slotframes. Indeed with 2 TX cells per slotframe, it takes 4 slotframes to see *MAX_NUMCELLS* cells. At the end of the estimation round, MSF takes a decision to allocate a new TX cell based on the ratio $\frac{NumCellsUsed}{NumCellsPassed} = \frac{8}{8} = 100\%$. The corresponding 6P ADD request is placed in front of the transmission queue, waiting the next TX cell to be sent. In the example, the delay before transmssion (*request delay*) is 2 slots. On the parent side, the 6P response is immediately placed on the TX queue. The 6P response spends 4 slots in the queue before being transmitted (*response delay*).

The duration of an *estimation round* changes with the current number of allocated cells. With k allocated cells, and assuming these are spread uniformly over the slotframe, one estimation round lasts $\frac{MAX\_NUMCELLS}{k}$ slotframes. We can derive the time to allocate multiple consecutive cells by summing the duration of consecutive rounds. The time $T_{\mathrm{msf}}(a,b)$ required by MSF to go from a to b allocated cells $(0 < a < b)$ can be expressed as shown in Eq. 2, with T_{sf} the slotframe duration.

$$T_{\mathrm{msf}}(a,b) = T_{\mathrm{sf}} \times \sum_{k=a}^{b-1} \frac{MAX\_NUMCELLS}{k} \tag{2}$$

The above model does not take into account the time required to send a 6P ADD request (*request delay*) and the time to receive a 6P response (*response delay*), let alone any packet loss and retransmission. Again, assuming cells are allocated uniformly over the slotframe, and taking into account that 6P messages are sent in priority, the expected time before transmission of a 6P request is $\frac{T_{\mathrm{sf}}}{2k}$ where k is the number of currently allocated TX cells. Assuming the reverse traffic is very light and the neighbor has allocated a single TX cell, the expected time before sending the 6P response is $\frac{T_{\mathrm{sf}}}{2}$. Taking these terms into account the model leads to Eq. (3).

$$T_{\mathrm{msf}}(a,b) = T_{\mathrm{sf}} \times \sum_{k=a}^{b-1} \left(\frac{1}{2} + \frac{1}{2k} + \frac{MAX\_NUMCELLS}{k}\right) \tag{3}$$

We now compare the times predicted by this model to those observed in the experiments with the linear topology and static traffic at a rate of 5pkts/sf (see Section 5.2). Figure 14 reports the occurence times of MSF decisions in 50 randomized runs of the experiments, for each node. The

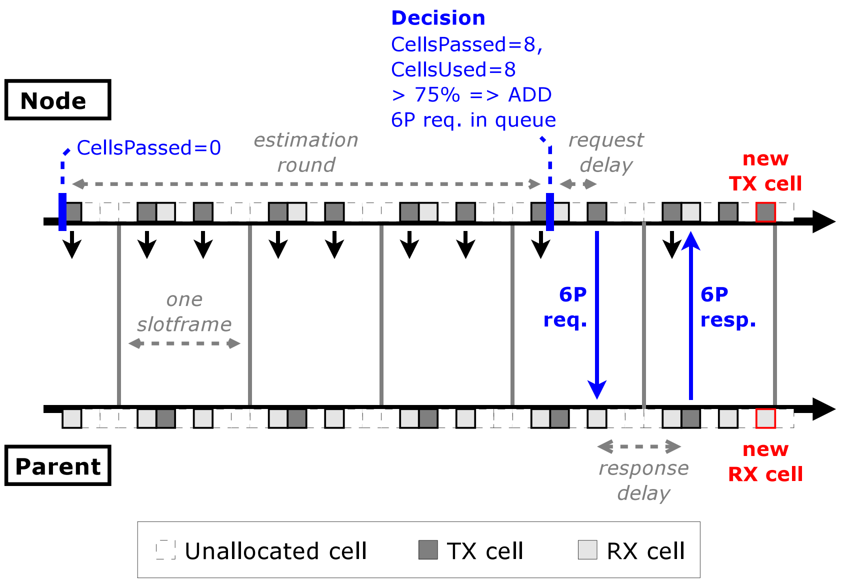


Figure 13 Duration of a single MSF allocation: traffic estimation, decision to allocate and 6P transaction.

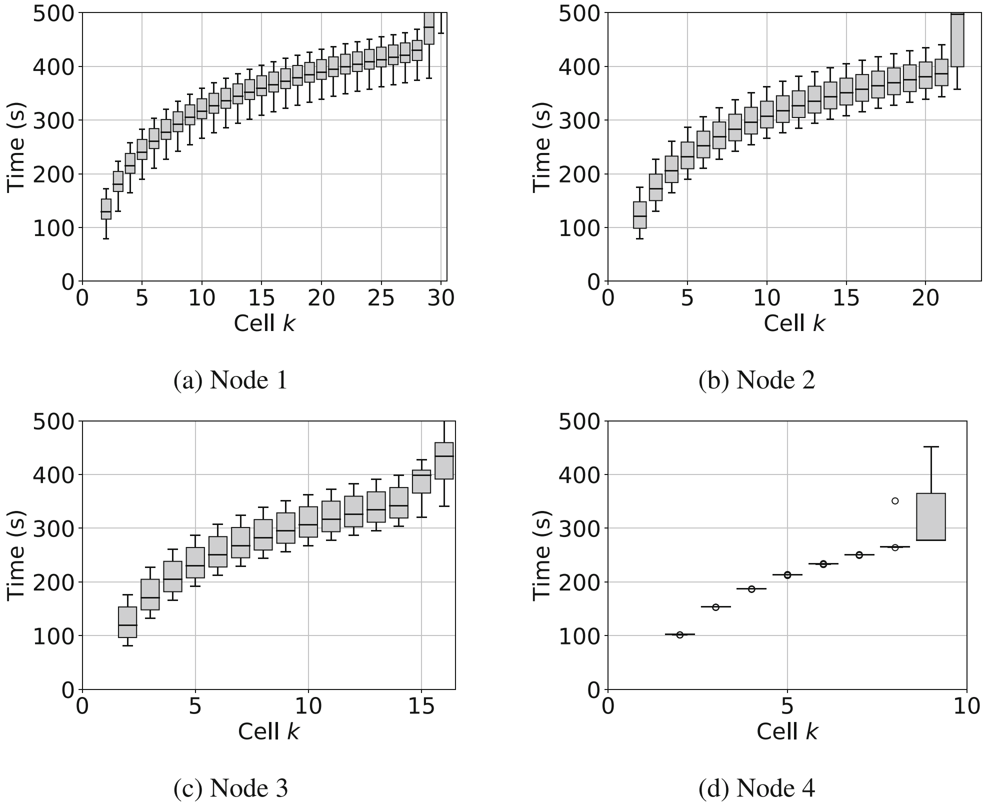


Figure 14 Time between application start and decision to allocate cell k. Each boxplot summarizes the distribution of this time across the 50 runs for a specific cell k. The y-axis is cut at 500s although there are a few higher values for the latest cell allocations.

x-axis is the number of TX cells allocated. The y-axis is time, $t = 0$s corresponding to when the application starts sending traffic. The slotframe initially contains a single TX cell which was installed upon network join. For each current slotframe allocation, the figure shows at what time the next allocation occurs (cell k). Since there are 50 runs, a summary of the distribution of this time is provided in the form of a boxplot. There is a lot of variance which is due to the first allocation.

The time to first allocation, i.e. to the 2nd cell allocation, is subject to a lot of variance across the 50 runs. On node 2, the median of this time is 120.6s but it extends from 79.3s to 175.2s. To understand that variance, we need to recall that the MSF estimator must be higher than *LIM_NUMCELLSUSED_HIGH* (75%) to allocate a new cell. However the beginning of an estimation round typically depends on when a node will join the network. We can consider this time as arbitrary. A decision is only taken after *MAX_NUMCELLS* cells have passed. Let's consider the two extreme situations depicted in Fig. 15. In this example, the value of *MAX_NUMCELLS* is 100. In the first situation, less than 25 slotframes have passed since the beginning of the estimation round, leaving at least 75 cells remaining to pass. All these cells are used by the traffic to carry data, hence at the end of the estimation round, MSF decides to allocate a new cell. On the contrary, in the second situation, more than 25 slotframes have passed when the application starts, hence the threshold cannot be exceeded at the end of the estimation round. In this case, MSF needs an additional round (100 slotframes) before it decides to allocate a new cell.

In the same experiments, we measured the *request delay*, the time between a decision to allocate cell k and the transmission of the corresponding 6P ADD request. Figure 16 reports those measurements, expressed in timeslots. A first observation is that this time is always below one slotframe, i.e. 101 timeslots. This is due to 6P packets always being put in front of the transmit queue. As shown in Eq. 3, our hypothesis is that the expected *request delay* varies as the inverse of the current number of allocated cells ($k - 1$). To this end, we also show on these figures bars for $\frac{101}{k-1}$.

Let's now turn to Fig. 17a and b to compare the experimental results to the predictions made by the model. To reduce the impact of variance across the runs, especially for the first allocation, we measured the inter-allocation times and show them relative to the median of the previous allocation on Fig. 17a. Moreover, all the times in Fig. 17b have been shifted by a correction offset $\Delta = 18.62s$ to account for the bias due to the unpredictable *estimation round* starting times. The value of Δ corresponds to the difference between the median of the measured 2nd cell allocation time, $\tilde{t_2} = 120.63s$, and the predicted time $T_{\text{msf}}(1, 2) = T_{\text{sf}} \times \left(\frac{1}{2} + \frac{1}{2} + 100\right) = 102.01s$. With that offset correction, the model and measurement fit pretty well visually up to cell 21.

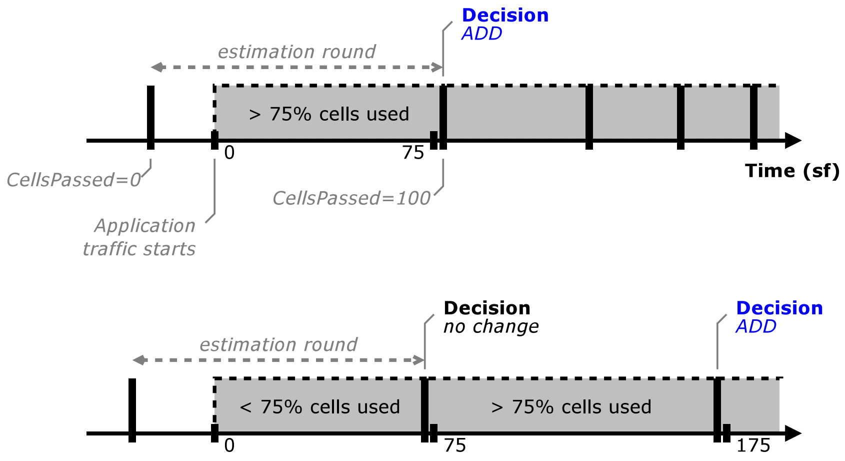


Figure 15 Variance in the occurence of MSF's first decision.

To further assess the validity of the model, we measured the error between the predicted times $T_{\mathrm{msf}}(1,k)$ and observed times t_k for all nodes over the 50 runs. To this end, we rely on two error measurements that we report on Fig. 18. First, the *absolute error* is the difference between observed and predicted times for the last cell added, that is $\epsilon = t_N - T_{\mathrm{msf}}(1,N)$, with N being the last cell considered. Second, the *root mean square error* (RMSE) between the predicted times and observed times for cells $2 \le k \le N$. The RMSE reports divergence measured during the whole allocations period. We exclude from the error the late cell allocations, i.e., the last cell and all cells that occur past 500s. The RMSE is calculated based on Eq. 4 and is reported relative to the convergence time. For example, for node 2, run 0, the time of the penultimate allocation (cell 22) is 347.8s and the RMSE is 4.07s, which gives a relative error of 1.17%. For the same node and run, the time predicted by the model for the penultimate allocation, with offset correction Δ, is 356s, an absolute error of $\epsilon = -8.2s$. There are a few runs with outliers for the absolute error (see Fig. 18a). A careful inspection of the runs in question reveals multiple consecutive late allocations. For example,

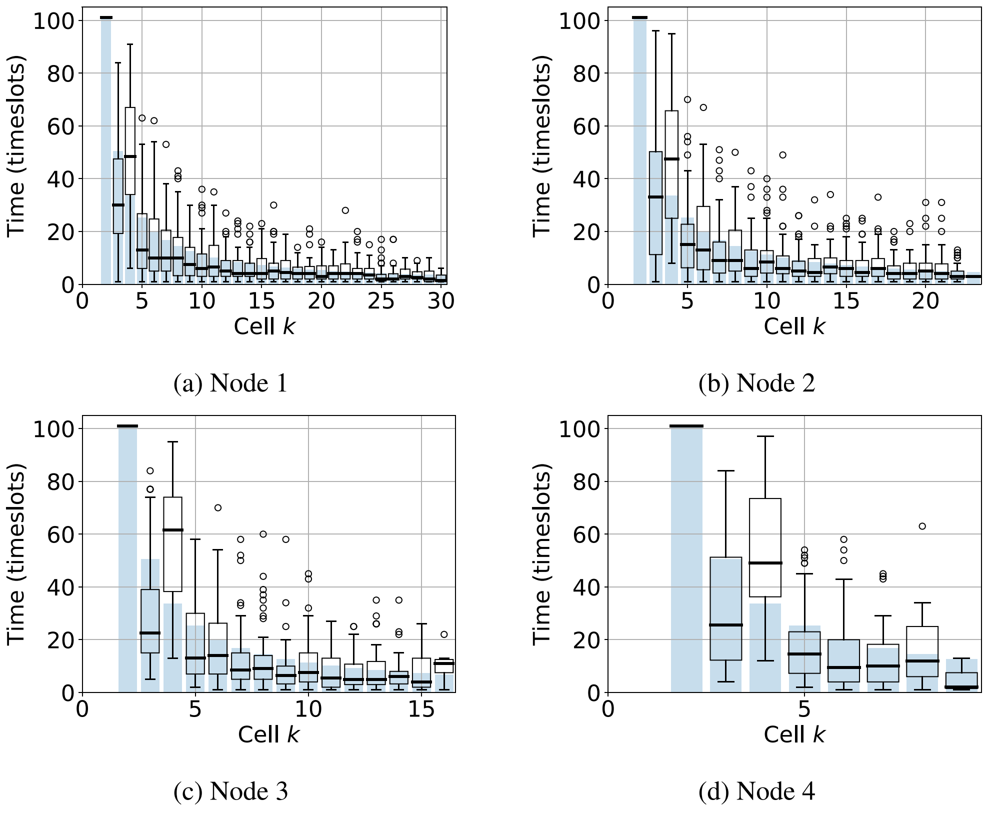
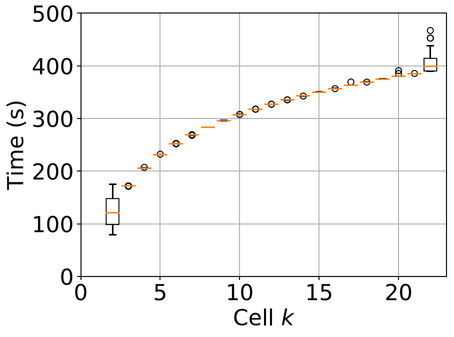
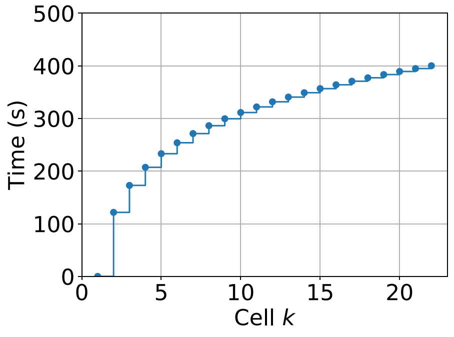


Figure 16 *Request delay*, time between decision to allocate cell k and transmission of the corresponding 6P ADD request, in timeslots. Each boxplot summarizes the distribution of this time across the 50 runs for a specific cell k. The blue bars correspond to $\frac{101}{k-1}$, the expected number of timeslots between two TX cells with $k-1$ cells currently allocated.



(a) Experiments



(b) Model $T_{\mathrm{msf}}(1,k)$

Figure 17 MSF convergence time for static traffic 5 pkts/sf, observed on node 2.

in run 28 on node 3, the last two allocations occur more than 100s later than the previous ones, but still within 500s from the application start.

$$\sqrt{\frac{1}{N-1}\sum_{k=2}^{N}(t_k - T_{\mathrm{msf}}(1,k))^2} \tag{4}$$

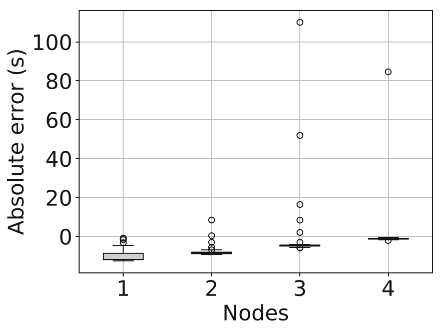
5.5 Link Between MSF Convergence and Losses

In Section 5.2, we observe some variability in the PDR performance (see e.g. Figure 6) across the 50 randomized runs and within each traffic rate. As an example, for the 1 pkt/sf rate, the median PDR is close to 92% but the range extends from 87% to almost 100%. Our first hypothesis was that this variability is caused by different convergence patterns of the MSF traffic adaptation mechanism. We thought that the variability observed also indicates that with similar initial conditions, MSF does not allocate resources at the same speed. These different convergence patterns may be caused by variation in the time required to complete 6P transactions to negotiate the cells in the schedule.

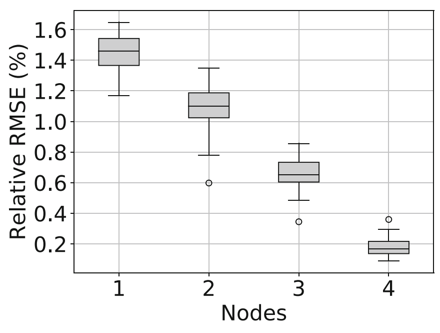
However, a more detailed investigation of the duration of 6P transactions reveals only a minor impact on the time required to allocate all the cells. Figure 19 shows the distribution of 6P ADD and DELETE transaction durations at different traffic rates for the constant traffic experiments.

The provided distributions span all the nodes and runs at a given traffic rate. One can observe that the median transaction duration is below one slotframe (101 timeslots). However, there is some variability. We do not show the outliers, that is the values that are beyond the 3rd quartile + 1.5 times the interquartile distance, in order to focus on the largest portion of the distribution. However, there is a significant number of outliers with 6P transaction durations that extend up to 1275 s. Such outliers are particularly important at the 10 packets/sf rate due to the fact that the traffic load on node 1 requires almost all the slotframe to be allocated, leading to many transactions initiated by node 2 ending with an `RC_ERR` error code ("*slotframe is full*"). However, the duration of the majority (99.96%) of transactions is below 2 slotframes, hence they have minimal impact on PDR.

Figure 20 compares the end-to-end PDR computed over three different periods, the PDR during the period of cell allocations, the global PDR over the complete simulation as presented in Fig. 6 and the PDR in steady-state after cell allocations. The low traffic rates at 0.1 and 0.2 pkt/sf do not cause any packet loss, hence a global PDR of 100% for those packet rates. For higher packet rates, we observe that the PDR during cell allocations is much worse than the global PDR. The end-to-end PDR in steady-state is always 100%. In other words, no loss occurs after cell allocations. This suggests that when traffic changes occur, the cell allocation



(a) Absolute error



(b) Relative RMSE

Figure 18 *Absolute error* and *root mean square error* (RMSE) between predicted and observed times, for each node and run. The RMSE is shown relative to the occurence time of the last cell allocation considered. Each boxplot summarizes the distribution of the errors over 50 runs.

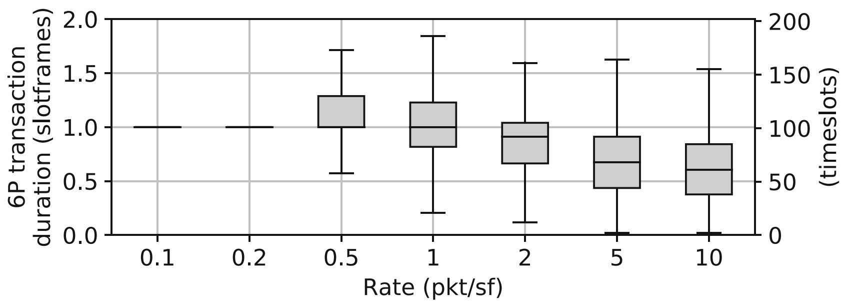


Figure 19 Duration of 6P transactions (excluding outliers).

periods result in a considerable amount of packet losses in addition to the losses caused by non-perfect links. When the TX queue is full, all new packets are dropped. It is only when resources are sufficient that the average emptying rate of the queue equals its filling rate and packets are not dropped anymore. Thus, the length of those allocation periods should directly impact the PDR of the network.

Figure 21 illustrates this behavior by showing the correlation between the time to the penultimate 6P ADD transaction (x-axis) and the number of queue overflow drops on the same node (y-axis) where queue drops means packets dropped because the TX queue is full. 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.

The lower PDR achieved at rate 10 pkts/sf is also explained by how close the slotframe size is to the theoretical number of required cells. On node 1, to carry 10 pkts/sf from child nodes along with its locally generated traffic, 1 shared cell, 40 TX cells and 30 RX cells are required, for a total of $N_{\text{req}} = 71$. Taking into account overprovisioning, this leads to at least $N_{\text{req}} Ovp = 95$ cells while the slotframe length is 101 cells. In practice, the number of cells requested is often higher than that, leading to the inability to satisfy all 6P requests. Figure 22a shows the distribution, over the different runs, of the number of times 6P ADD requests failed due to the parent's slotframe being full. Most failures occur on node 2 which is unable to get all its requests fulfilled by node 1. Figure 22b shows that these errors can span a long period of time, with continuous requests being made to the parent without success. This also explains the origin of the span of the cluster for the 10 pkts/sf rate in Fig. 21. Retries of failed requests delay the occurrence of the penultimate 6P ADD transaction up to the end of the simulation.

Figure 21 shows that the number of losses correlates with the length of the cell allocations period. We speculate that the variation in duration for the cell allocations period results from variance in the occurence of the first MSF decisions and cell allocations (see Section 5.4). Figure 23 extends this correlation between the occurence of MSF's first decision and the number of queue overflow drops observed for packet rates 2 – 10 pkts/sf.

6 Modification of MSF Parameters

In Section 5, we observed that the convergence pattern of MSF has a large impact on the number of packet losses and usage of resources. On one hand, we observed that

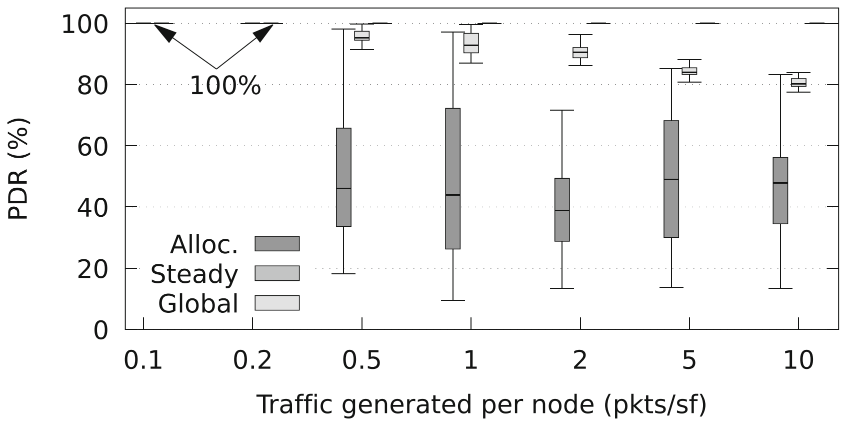


Figure 20 Comparison of end-to-end PDR during different periods of the simulation.

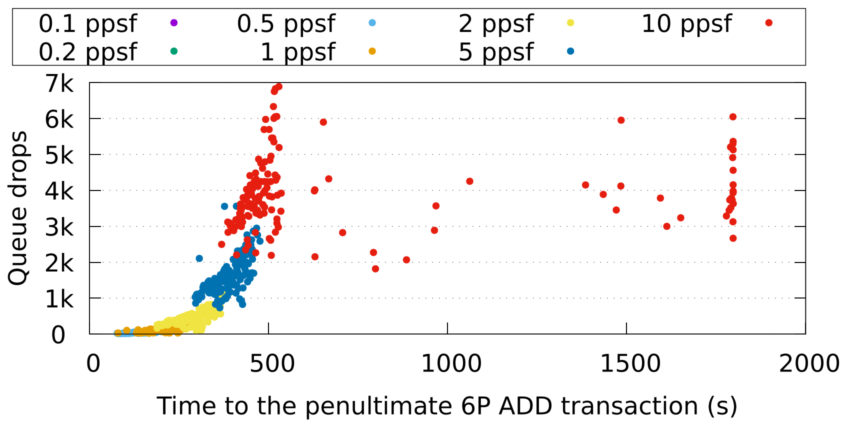
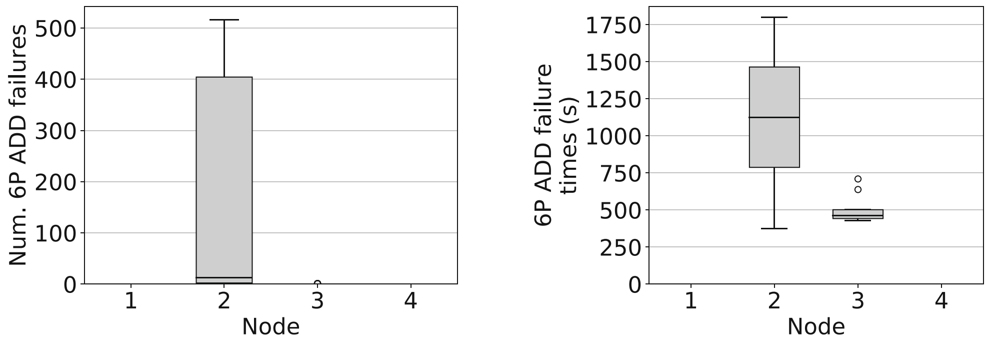
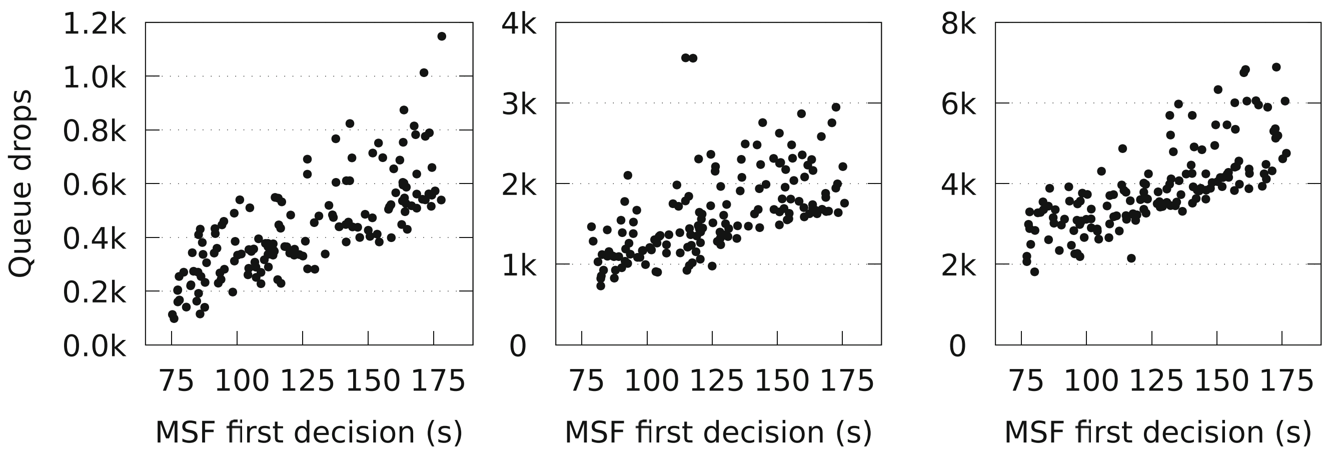


Figure 21 Correlation between the penultimate 6P ADD transaction and the number of losses observed for 50 runs, all nodes and all packet rates. The color distinguishes the packet rates. The time to the penultimate 6P ADD transaction is the time required for MSF to allocate the necessary resources. We observe that the number of queue overflow drops increases with the time to allocate those resources.



(a) Number of failures (b) Failure times

Figure 22 6P ADD failures at rate 10 pkts/sf. **a** shows the number of 6P ADD requests initiated by each node that resulted in a failure because the parent slotframe is full and thus cannot fulfill the request. **b** shows, with the same failures, the time required by each node to complete the 6P ADD transactions and have the cells allocated in its slotframe.



(a) 2 pkts/sf. (b) 5 pkts/sf. (c) 10 pkts/sf.

Figure 23 Correlation between the occurence of MSF's first decision and the number of queue overflow drops. The time to MSF's first decision can vary significantly from one run to another, delaying resource allocation and resulting in more packet losses.

MSF allocates new resources at a rate which depends on the number of the already allocated cells, i.e., with the slowest rate occurring when the slotframe is mostly empty and higher rates when the slotframe already contains allocated cells. During this allocation process, packets exceeding the current transmission capacity are enqueued and dropped when the queue is full.

On the other hand, MSF also leads to overprovisioning, i.e., more cells are allocated than what is actually required to carry the current traffic load. Indeed, moderate overprovisioning is beneficial as it allows to quickly absorb small, transient increases in traffic load without triggering new cell allocations. However, as shown in Fig. 10, even with a constant rate traffic, the amount of overprovisioning in MSF is significant, thus wasting a large part of the available network resources.

In this section, we show the impact of changing implementation constants on the convergence pattern of the scheduling function.

6.1 Towards Faster Resource Allocation

To speed up resource allocation, we can reduce the *MAX_NUMCELLS* constant. Indeed, this value represents the length (in number of elapsed cells) of the window used to estimate the current traffic load. Reducing the size of this window would reduce the time between a change in traffic load and the decision to allocate more or free existing resources, thus speeding up the convergence. In fact, this idea has also been explored recently by [4].

However, reducing *MAX_NUMCELLS* has a side effect: it makes the estimation of the current traffic load less precise as it is computed on a shorter sample. Moreover, the granularity of the estimation is given by $\frac{1}{MAX\_NUMCELLS}$. A coarser granularity and a less accurate estimation of the load can trigger unexpected 6P transactions, and even oscillations.

To further investigate the effect of *MAX_NUMCELLS* on the behavior of MSF, we reproduce the experiments of Section 5.3 with changing traffic. However, this time we use different values of *MAX_NUMCELLS*, both higher and lower than the default value. In this set of experiments, we increase the traffic rate from 0 pkt/sf to 5pkts/sf, then to 10 pkts/sf, then we reduce to 5 pkts/sf, and finally back to 0 pkt/sf. Figure 26 reports the evolution of the resource allocation period for three values of the *MAX_NUMCELLS*: 25, 100 (default) and 200, while Table 2 summarizes the extent of the first and second allocation period.

As expected, the convergence time is almost directly proportional to *MAX_NUMCELLS*. Indeed, switching from a window of size 100 (Fig. 24b) to 25 (Fig. 24a) reduces the time to allocate the resources from 250.46 s to 71.69 s during the first convergence period. On the contrary, by increasing the *MAX_NUMCELLS* to 200 (Fig. 24c) extends the convergence time of this period to 497.91 s.

Furthermore, even though the traffic is purely periodic with perfect link quality (PDR=100%), by reducing the *MAX_NUMCELLS*, the variance of the MSF TX estimator increases. To explain why this occurs, we employ the toy example depicted in Fig. 25. It shows a slotframe of length 17 timeslots, with constant traffic and *MAX_NUMCELLS*=5. The gray cells are the allocated ones, while the cells marked with a “U” are the used ones. Then, two estimation periods are depicted. The first period extends from TSN 15 to TSN 6, where it takes 9 timeslots to get 5 allocated cells of which 4 are used, leading thus to a traffic load estimation of 80%. The second period extends from TSN 7 to TSN 14, taking 8 timeslots to reach 5 allocated cells of which a single one is used, leading thus to an estimation of 20%. If that pattern is recurring, then the estimation will oscillate with a value above the high 75% threshold and a value below the low 25% threshold, and, thus, potentially triggering constant allocation and deallocation.

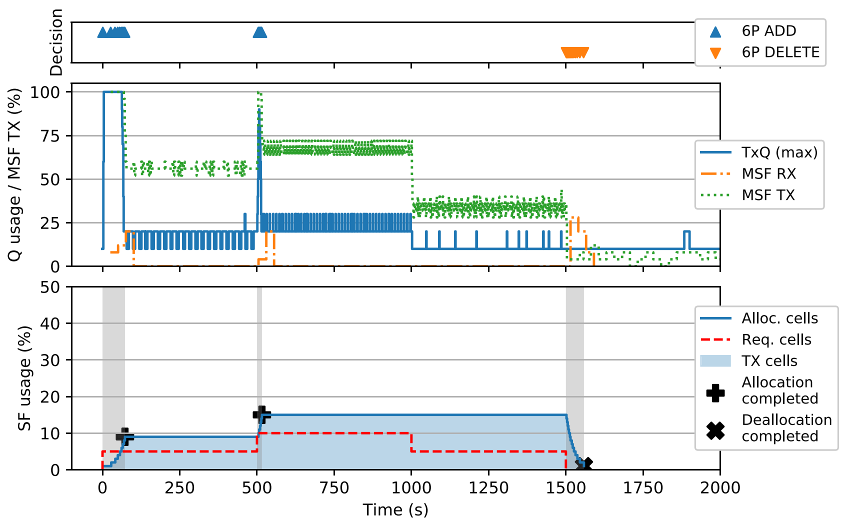
6.2 Limiting Overprovisioning

To limit the impact of overprovisioning, one could try to modify the thresholds based on which MSF decides to allocate or deallocate cells. These values, known as *LIM_NUMCELLSUSED_HIGH* and *LIM_NUMCELLSUSED_LOW*, have default values fixed at 75% and 25%, respectively. As soon as the MSF estimator reaches 75%, the allocation of a new cell is triggered. Similarly, when the estimator goes below 25%, the deallocation of an existing cell is triggered.

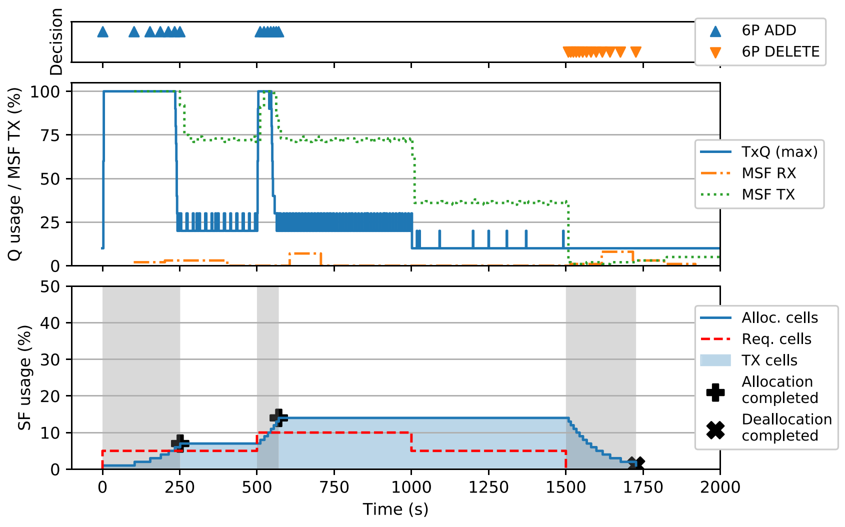
Table 2 Duration and amplitude of the first (0 s $\rightarrow$ 500 s) and second (500 s $\rightarrow$ 1000 s) allocation periods along with the theoretical duration predicted by the convergence model.

	1st period ($t = 0$s)			2nd period ($t = 500$s)		
MAX_NUMCELLS	Alloc.	Dur. (s)	Theo. (s)	Alloc.	Dur. (s)	Theo. (s)
25	1 → 9	71.69	74.03	9 → 15	15.08	16.78
100	1 → 7	250.46	251.71	7 → 14	69.62	77.64
200	1 → 7	497.91	499.17	7 → 14	145.37	151.39

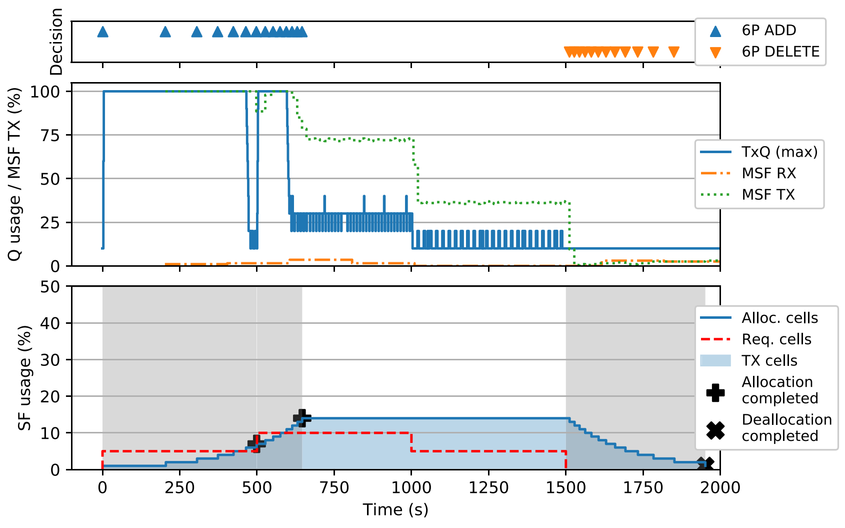
Figure 24 Evolution of allocated cells for different values of *MAX_NUMCELLS*. Large values of *MAX_NUMCELLS* result in a long convergence time. Lower values of *MAX_NUMCELLS* increase the variance on MSF TX.



(a) MAX_NUMCELLS = 25.



(b) MAX_NUMCELLS = 100 (default).



(c) MAX_NUMCELLS = 200.

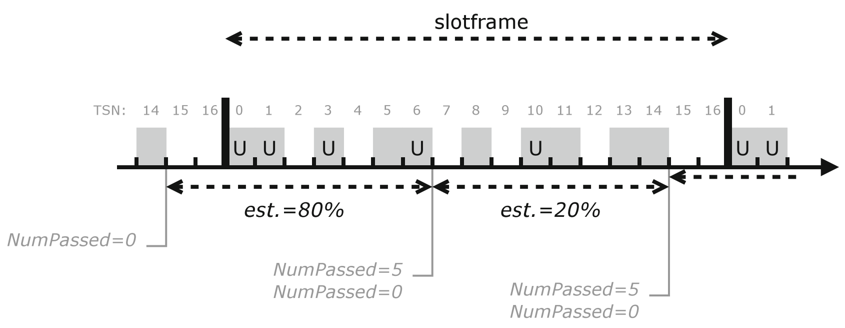


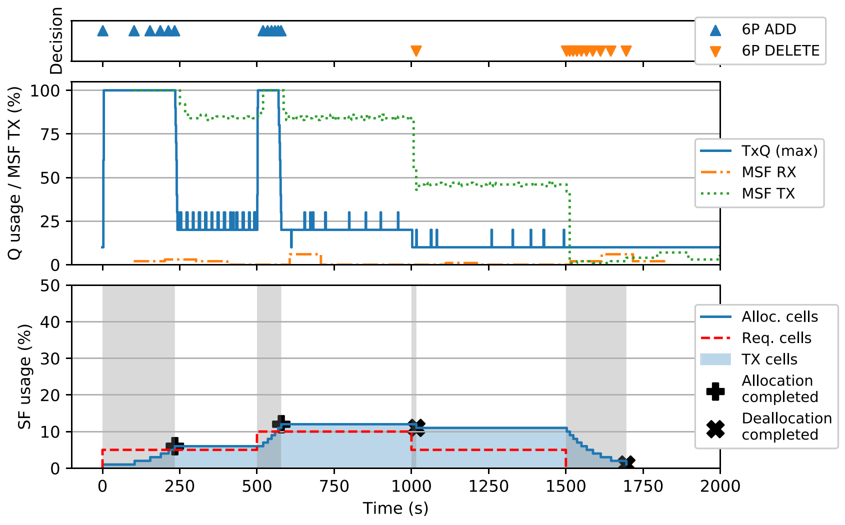
Figure 25 Effect of small *MAX_NUMCELLS* values.

The use of different thresholds prevents a rapid reaction in case of small fluctuations of the decision variable, a principle often referred to as *hysteresis*.

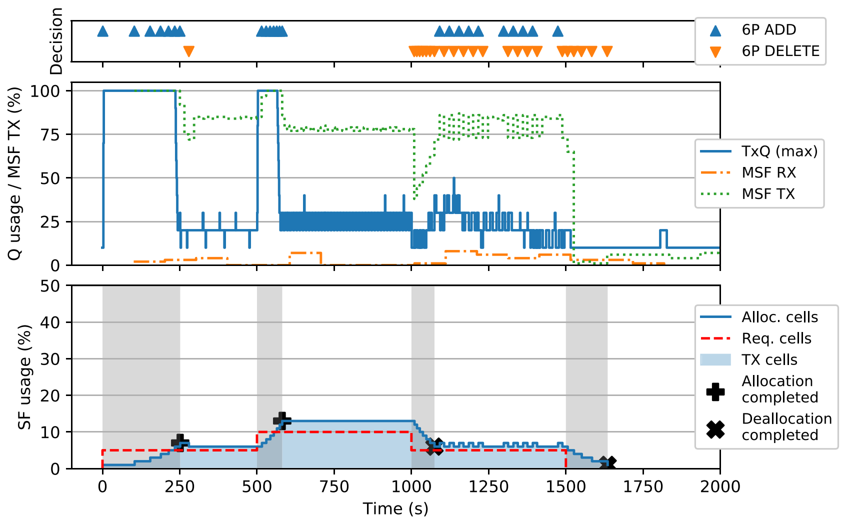
In Section 5, we observed that MSF often leads to overprovisioning. Hereafter, we distinguish two types of overprovisioning.

1. **Overprovisioning due to allocations:** using a high threshold of 75% means that new allocations continue until the current number of used cells is no more than 75% of the allocated cells. This approach leads to overprovisioning, in which the amplitude can be estimated as $\frac{1}{75\%} = 133\%$.

Figure 26 Evolution of allocated cells for different values of the MSF thresholds. A high threshold value reduces overprovisioning due to allocations, but is still subject to overprovisioning due to missed deallocations. Close threshold values limit overprovisioning but trigger constant allocations/deallocations.



(a) Thresholds of 45% and 95%.



(b) Thresholds of 75% and 85%.

2. **Overprovisioning due to missed deallocations:** when the traffic load decreases MSF deallocates only if the current number of used cells is below 25% of the allocated cells. If the number of used cells decreases to this limit but not below 25%, the amplitude of overprovisioning can reach up to 400% of the required amount of cells.

We can draw from the above observations that a high threshold value closer to 100% should reduce overprovisioning due to allocations while using a higher value for the low threshold should allow MSF to deallocate faster. Hereafter, we investigate how changing these thresholds affects the behavior of MSF.

Figure 26 shows the impact of changing the values of the resource usage thresholds with the default value of $MAX\_NUMCELLS = 100$. Figure 24b shows the default behavior (25% and 75%). In Fig. 26a, we adapted those thresholds to above 95% to allocate new cells and below 45% to deallocate cells. This will result in overprovisioning going from $\frac{1}{75\%} = 133\%$ to $\frac{1}{95\%} = 105\%$ of the theoretical number of cells required. As we can observe in Fig. 26a, during the two first periods of resource allocations at $t > 250$ s and $t > 500$ s, the number of cells allocated (the black line) is closer to the theoretical number of required cells (the dashed line) than it is in Fig. 24b.

Furthermore, we see at $t > 1000$ s that as the traffic decreases, so does the estimation of the resource usage by MSF. Although it triggers a single deallocation, we reach slightly below 45%, and no additional cells are removed from the schedule. As a result, the amount of overprovisioning, i.e., the difference from the theoretical number of required cells, increases.

One could think that the solution to this problem would be to raise the threshold in order to start the deallocations earlier. In that case, the closer the high and low thresholds are, the faster MSF can trigger cell deallocations. For example, we can take a high threshold of 85% and a low threshold of 75%, as shown in Fig. 26b. As it can be observed, although we deallocate earlier, and, thus reduce both types of overprovisioning, this triggers a nearly constant cycle of allocation and deallocation along with large variations in the estimation of resource usage by MSF.

7 Conclusions

The deployment of Industrial IoT networks requires that they can quickly adapt to traffic changes in interference-prone environments. The Minimal Scheduling Function (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 provide a detailed description of MSF traffic estimation and cell allocation processes. We then employed the 6TiSCH simulator to evaluate the ability of MSF to allocate the required network resources in simple controlled experiments with constant and varying traffic loads. We observed that with retransmissions disabled but no interferences, packet losses can appear as soon as the traffic adaptation mechanism becomes necessary. This is to be expected considering the reactive nature of MSF.

However, 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 it depends on the number of cells already allocated in the slotframe. To deepen our understanding of this allocation process, we derived a mathematical model of the MSF convergence time that we then compared to our experimental results. 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 is reluctant to release cells that it previously allocated.

Finally, in order to assess the degree of customizability offered by MSF, we then evaluate its behaviour when varying its main parameters with two objectives in mind: reduce the convergence time and limit the amount of over-provisioning. We come to the conclusion that none of these approaches work reliably, as they can easily lead the protocol to unstable behaviour. From these results, it appears that improving MSF would require a modification of the scheduling function itself. For instance, the proposed A-MSF variant [4] aims to speed up the convergence of the network with the possibility to allocate multiple cells per MSF decision. However, this approach requires a more careful evaluation. Indeed, allocating multiple cells on each decision could exacerbate the problem of over-provisioning, which is already significant with the current version of MSF allocating only one cell at a time.

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.

Authors must disclose all relationships or interests that could have direct or potential influence or impart bias on the work:

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. IEEE Standard for Low-Rate Wireless Networks (2016). *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, 1–709. https://doi.org/10.1109/IEEESTD.2016.7460875.
2. Accettura, N., Vogli, E., Palattella, M., Grieco, L., Boggia, G., Dohler, M. (2015). Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation. *IEEE Internet of Things Journal (IoT-J), 2*. https://doi.org/10.1109/JIOT.2015.2476915.
3. Bahl, P., Chandra, R., Dunagan, J. (2004). SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking* (pp. 216–230). ACM.
4. Chang, T., Vučinić, M., Vilajosana, X., Dujovne, D., Watteyne, T. (2020). 6TiSCH Minimal Scheduling Function: Performance Evaluation. *Internet Technology Letters, 3*. https://doi.org/10.1002/itl2.170.
5. Chang, T., Vučinić, M., Vilajosana, X., Duquennoy, S., Dujovne, D. (2020). 6TiSCH Minimal Scheduling Function (MSF). Internet-Draft draft-chang-6tisch-msf-18, Internet Engineering Task Force. https://tools.ietf.org/html/draft-ietf-6tisch-msf-18. Work in Progress.
6. Chang, T., Watteyne, T., Wang, Q., Vilajosana, X. (2016). LLSF: Low Latency Scheduling Function for 6TiSCH Networks. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)* (pp. 93–95). https://doi.org/10.1109/DCOSS.2016.10.
7. Daneels, G., Spinnewyn, B., Latré, S., Famaey, J. (2018). ReSF: Recurrent Low-Latency Scheduling in IEEE 802.15.4e TSCH networks. *Ad Hoc Networks 69*, 100–114. https://doi.org/10.1016/j.adhoc.2017.11.002. http://www.sciencedirect.com/science/article/pii/S1570870517302019.
8. Domingo Prieto, M., Chang, T., Vilajosana, X., Watteyne, T. (2016). Distributed PID-based Scheduling for 6TiSCH Networks. *IEEE Communications Letters, 20*. https://doi.org/10.1109/LCOMM.2016.2546880.
9. Dujovne, D., Grieco, L.A., Palattella, M.R., Accettura, N. (2015). 6TiSCH On-the-Fly Scheduling. Internet-Draft "draft-dujovne-6tisch-on-the-fly-06.txt", *Internet Engineering Task Force*. http://www.ietf.org/id/draft-dujovne-6tisch-on-the-fly-06. (Work in progress).
10. Duquennoy, S., Al Nahas, B., Landsiedel, O., Watteyne, T. (2015). Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *Proceedings of the 13th ACM conference on embedded networked sensor systems* (pp. 337–350). ACM.
11. Duquennoy, S., Elsts, A., Nahas, A., Oikonomou, G. (2017). TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2015)*.
12. Duquennoy, S., Eriksson, J., Voigt, T. (2017). Five-nines reliable downward routing in RPL. CoRR arXiv:http://arxiv.org/abs/1710.023241710.02324.
13. Guglielmo, D.D., Anastasi, G., Seghetti, A. (2014). From IEEE 802.15.4 to IEEE 802.15.4e: A step towards the Internet of Things. *Advances in Intelligent Systems and Computing*.
14. Hamza, T., & Kaddoum, G. (2019). Enhanced Minimal Scheduling Function for IEEE 802.15.4e TSCH Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*.
15. Hauweele, D., Koutsiamanis, R., Quoitin, B., Papadopoulos, G.Z. (2020). Pushing 6TiSCH Minimal Scheduling Function (MSF) to the Limits. In *IEEE Symposium on Computers and Communications, ISCC 2020* (pp. 1–7). Rennes: IEEE. https://doi.org/10.1109/ISCC50000.2020.9219692.
16. Hermeto, R.T., Gallais, A., Theoleyre, F. (2017). Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial networks: A survey. *Computer Communications, 114*, 84–105.
17. Jeong, S., Kim, H.S., Paek, J., Bahk, S. (2020). OST: On-Demand TSCH Scheduling with Traffic-Awareness. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications* (pp. 69–78). https://doi.org/10.1109/INFOCOM41043.2020.9155496.
18. Kim, S., Kim, H., Kim, C. (2019). ALICE: Autonomous Link-based Cell Scheduling for TSCH. In *2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (pp. 121–132).
19. Kotsiou, V., Papadopoulos, G.Z., Chatzimisios, P., Theoleyre, F. (2020). LDSF: Low-Latency Distributed Scheduling Function for Industrial Internet of Things. *IEEE Internet of Things Journal, 7*(9), 8688–8699. https://doi.org/10.1109/JIOT.2020.2995499.
20. Koutsiamanis, R.A., Papadopoulos, G.Z., Fafoutis, X., Fiore, J.M.D., Thubert, P., Montavont, N. (2018). From Best-Effort to Deterministic Packet Delivery for Wireless Industrial IoT Networks. *IEEE Transactions on Industrial Informatics, 14*, 4468–4480.
21. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D. (2007). Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC 4944 (Draft Standard). http://www.ietf.org/rfc/rfc4944.txt.
22. Municio, E., Daneels, G., Vučinić, M., Latré, S., Famaey, J., Tanaka, Y., Brun, K., Muraoka, K., Vilajosana, X., Watteyne, T. (2018). Simulating 6TiSCH networks. *Transactions on Emerging Telecommunications Technologies*.
23. Palattella, M.R., Accettura, N., Dohler, M., Grieco, L.A., Boggia, G. (2012). Traffic Aware Scheduling Algorithm for Reliable Low-Power Multi-Hop IEEE 802.15.4e Networks. In *Proceedings of the 23rd IEEE International Symposium on Personal, Indoor and Mobile Radio Communicarions (PIMRC)* (pp. 327–332).
24. Palattella, M.R., Watteyne, T., Wang, Q., Muraoka, K., Accettura, N., Dujovne, D., Grieco, L.A., Engel, T. (2016). On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks. *IEEE Sensors Journal, 16*, 550–560.
25. Wang, Q.X., & Vilajosana, T.W. (2018). 6TiSCH Operation Sublayer (6top) Protocol (6P). Tech. Rep. 8480, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc8480.txt.
26. Righetti, F., Vallati, C., Anastasi, G., Das, S. (2018). Analysis and Improvement of the On-The-Fly Bandwidth Reservation Algorithm for 6TiSCH. In *Proceedings of the 19th International IEEE Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)* (pp. 1–9). https://doi.org/10.1109/WoWMoM.2018.8449793.
27. Righetti, F., Vallati, C., Das, S.K., Anastasi, G. (2020). An Evaluation of the 6TiSCH Distributed Resource Management Mode. ACM Trans. Internet Things **1**(4). https://doi.org/10.1145/3395927.
28. Shelby, Z., Hartke, K., Bormann, C. (2014). The Constrained Application Protocol (CoAP). RFC 7252. https://doi.org/10.17487/RFC7252. https://rfc-editor.org/rfc/rfc7252.txt.
29. Thubert, P. (2019). An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4. Internet-Draft draft-ietf-6tisch-architecture-28, Internet Engineering Task Force. https://tools.ietf.org/html/draft-ietf-6tisch-architecture-28. Work in Progress.
30. Thubert, P., Watteyne, T., Palattella, M.R., Vilajosana, X., Wang, Q. (2013). IETF 6TSCH: Combining IPv6 connectivity with industrial performance. In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)* (pp. 541–546). IEEE.

31. Valdovinos, I.A., Millán, P.E.F., Pérez-Díaz, J.A., Vargas-Rosales, C. (2021). Distributed Channel Ranking Scheduling Function for Dense Industrial 6TiSCH Networks. *MDPI Sensors, 5*. https://doi.org/10.3390/s21051593.
32. Vilajosana, X., Pister, K., Watteyne, T. (2017). Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration. RFC 8180. https://doi.org/10.17487/RFC8180. https://rfc-editor.org/rfc/rfc8180.txt.
33. Vilajosana, X., Watteyne, T., Chang, T., Vučinić, M., Duquennoy, S., Thubert, P. (2020). IETF 6TiSCH: A Tutorial. *IEEE Communications Surveys & Tutorials, 22*, 595–615.
34. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R. (2012). RPL: IPv6 routing protocol for low power and lossy networks. RFC 6550 (Draft Standard). http://www.ietf.org/rfc/rfc6550.txt.