This article has been accepted for publication in ACM Trans. Internet Things.
This is the author's version which has not been fully edited and content may change prior to final publication.
Citation information: DOI 10.1145/3395927

An Evaluation of the 6TiSCH Distributed Resource Management Mode

FRANCESCA RIGHETTI, Dept. of Information Engineering, University of Pisa, Italy
CARLO VALLATI, Dept. of Information Engineering, University of Pisa, Italy
SAJAL K. DAS, Dept. of Computer Science, Missouri University of Science and Technology, USA
GIUSEPPE ANASTASI, Dept. of Information Engineering, University of Pisa, Italy

The IETF is currently defining the 6TiSCH architecture for the Industrial Internet of Things to ensure reliable and timely communication. 6TiSCH relies on the IEEE TSCH MAC protocol and defines different scheduling approaches for managing TSCH cells, including a distributed (*neighbor-to-neighbor*) scheduling scheme, where cells are allocated by nodes in a cooperative way. Each node leverages a *Scheduling Function (SF)* to compute the required number of cells, and the 6top (6P) protocol to negotiate them with neighbors. Currently, the *Minimal Scheduling Function (MSF)* is under consideration for standardization. However, multiple SFs are expected to be used in real deployments, in order to accommodate the requirements of different use cases. In this paper, we carry out a comprehensive analysis of 6TiSCH distributed scheduling to assess its performance under realistic conditions. Firstly, we derive an analytical model to assess the 6P protocol, and we show that 6P transactions take a long time to complete and may also fail. Then, we evaluate the performance of MSF and other distributed SFs through simulations and real experiments. The results show that their performance is affected by the failure of 6P transactions and the instability of the routing protocol, which may lead to congestion from which the network is unable to recover. Finally, we propose a new SF (E-OTF) and show, through simulations and real experiments, that it can effectively improve the overall performance, by allowing nodes to quickly recover from congestion.

CCS Concepts: • **Networks** → **Network protocols**; **Network performance analysis**.

Additional Key Words and Phrases: 6TiSCH, 6top, Distributed Scheduling Function, MSF, OTF, RPL

ACM Reference Format:
Francesca Righetti, Carlo Vallati, Sajal K. Das, and Giuseppe Anastasi. 2018. An Evaluation of the 6TiSCH Distributed Resource Management Mode. *ACM Trans. Internet Things* 1, 1, Article 1 (January 2018), 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In industrial environments, Wireless Sensor Networks (WSNs) are a crucial enabling technology to reduce costs and improve efficiency of industrial processes. In this context, a reliable and timely communication is required to satisfy the needs of critical applications, such as remote control and

A preliminary version of this work appeared in the IEEE 19*th* International Symposium on "A World of Wireless, Mobile and Multimedia Network" (WoWMoM 2018).
Authors' addresses: Francesca Righetti, Dept. of Information Engineering, University of Pisa, Via Girolamo Caruso 16, Pisa, Italy, francesca.righetti@ing.unipi.it; Carlo Vallati, Dept. of Information Engineering, University of Pisa, Via Girolamo Caruso 16, Pisa, Italy, carlo.vallati@unipi.it; Sajal K. Das, Dept. of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA, sdas@mst.edu; Giuseppe Anastasi, Dept. of Information Engineering, University of Pisa, Via Girolamo Caruso 16, Pisa, Italy, giuseppe.anastasi@unipi.it.

2577-6207/2018/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

monitoring. To this end, in 2015 the IEEE issued a revised version of the popular 802.15.4 standard for WSNs [1]. The revised version includes the *Time Slotted Channel Hopping (TSCH)* Medium Access Control (MAC) protocol that specifically targets industrial and embedded applications. It relies on *time-slotted* access, *multi-channel communication* and *frequency hopping*, and can ensure guaranteed bandwidth, deterministic latency, reliable communication, high aggregate throughput, and energy efficiency.

The evolution towards the *Industrial Internet of Things (IIoT)* will require the interconnection of WSNs with IPv6 networks, in order to allow the seamless integration with existing platforms and services. To foster the integration of TSCH WSNs into the IPv6 infrastructure, IETF recently formed the"IPv6 over the TSCH mode of IEEE 802.15.4e" (6TiSCH) Working Group (WG). This WG is currently defining the 6TiSCH architecture [30, 35] to allow the adoption of IPv6-based protocols, such as 6LoWPAN and IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), on top of the TSCH MAC protocol.

Since TSCH does not specify how communication resources (i.e., TSCH cells) are allocated to nodes, the 6TiSCH architecture introduces an additional layer above the MAC layer to provide the concept of IPv6 link over TSCH and define the policy to manage TSCH cells. 6TiSCH considers four different paradigms for building and maintaining the TSCH schedule [30], namely *static*, *centralized*, *distributed (neighbor-to-neighbor)*, and *hop-by-hop scheduling*. However, the distributed approach has attracted more attention, as it guarantees self-configuration and adaptability to time-varying network conditions [27]. In distributed scheduling, a Scheduling Function (SF) is used by each node to calculate dynamically the number of cells to allocate, while the 6top (6P) protocol is used to negotiate the required cells with its neighbors [36]. Currently, the *Minimal Scheduling Function (MSF)* [4] is considered by the 6TiSCH WG for standardization as reference SF, while previously, the *On-The-Fly* (OTF) algorithm [27] was considered. However, multiple SFs are expected to be used in practice in order to meet the requirements of different use cases [30]. Hence, many different SFs have been proposed in the literature [2–5, 8, 16, 25, 27].

In this work, we carry out a comprehensive analysis of the 6TiSCH distributed mode, to assess its performance, mainly in terms of *end-to-end reliability* and *latency* provided to the application, under realistic conditions. In our study, we consider not only the SF, but also its interplay with the 6P and RPL protocols.

First, we analyze the performance of the 6P protocol by developing an analytical model, based on a Markov-Chain, to investigate the dynamics of 6P transactions. This allows us to obtain a raw estimation of the time required to complete a resource negotiation between neighboring nodes, and its success probability. Although some previous works assume instantaneous 6P transactions [24, 27], our analytical results show that, when considering non-ideal wireless links, a 6P negotiation may take a considerable amount of time to complete and its failure probability is not negligible.

Then, we carry out a performance evaluation of the overall 6TiSCH distributed mode. To this end, we consider three different SFs taking a different approach to cell allocation and deallocation, and investigate the impact on their performance of the 6P and RPL protocols. Specifically, we analyze MSF [4], OTF [27], and PID [8]. The evaluation is carried out through simulations and experiments on a real testbed. Our analysis show that the performance of the SF is significantly affected by route changes, triggered by RPL, and scheduling mismatches caused by the failure of 6P transactions. These events may cause congestion at intermediate nodes along the path towards the destination, resulting in high end-to-end latency and low reliability.

In order to overcome these issues, we propose and evaluate a new SF, named *Enhanced-OTF* (E-OTF). By introducing mechanisms to react to congestion and quickly recover from it, the proposed solution proves to be effective in improving the overall performance. The results obtained show that, in comparison with MSF, E-OTF provides a higher reliability, especially at high traffic rates

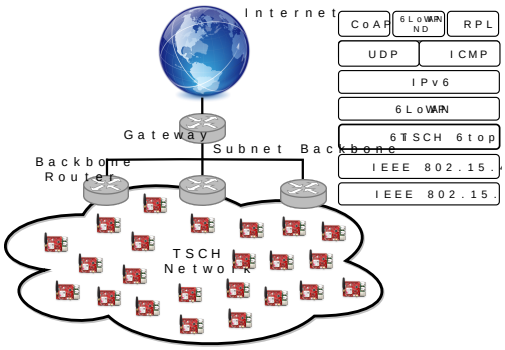


Fig. 1. The 6TiSCH Architecture.

(90% vs. 65%, when the traffic rate is 2 packets per slotframe), and reduces the end-to-end latency (up to 67%). The gain in performance is even better with respect to OTF and PID.

Although other works have assessed the performance of (distributed) SFs for 6TiSCH, to the best of our knowledge, this is the first work analyzing the performance of MSF (the SF currently considered for standardization) through simulation and measurements on a real testbed. In addition, it is the first work analyzing the performance of 6TiSCH distributed scheduling in a comprehensive way. Our study provides an insight on the influence of the routing protocol and the dynamics of 6P negotiations on the performance of the SF under realistic conditions.

In a nutshell, the main contributions made by this paper can be summarized as follows:

- An analytical model to evaluate the performance of the 6P protocol under lossy links;
- A performance evaluation of MSF under realistic conditions;
- A comprehensive evaluation of the 6TiSCH distributed mode to investigate the impact of the 6P and RPL protocols on the SF;
- A new SF for 6TiSCH;
- An experimental analysis on a testbed to validate the simulation results in a real environment.

The remainder of the paper is organized as follows. Section 2 introduces the 6TiSCH architecture. Section 3 surveys the related work. Section 4 presents an analytical model for 6P transactions. Section 5 introduces the three SFs (MSF, OTF and PID) considered in our performance evaluation, whose results are discussed in Section 6. Section 7 describes the proposed E-OTF SF, while Section 8 shows the results of its performance comparison with MSF and OTF. Section 9 describes the testbed used to validate the simulation results and discusses the experimental results. Finally, Section 10 concludes the paper.

2 6TISCH ARCHITECTURE

The 6TiSCH architecture [30] aims at integrating TSCH networks into IPv6 networks. The overall architecture and the complete protocol stack are depicted in Fig. 1. At the MAC layer, the TSCH MAC protocol ensures deterministic latency, high reliability, and low energy consumption. Unlike the traditional *Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)* algorithm used in the original 802.15.4 standard [1], TSCH leverages a time-slotted channel-access mode. Time is divided into fixed-size intervals, called *timeslots*, grouped into *slotframes* that repeat periodically over time. To increase the network capacity, concurrent transmissions by different nodes on the same timeslot are allowed, using different channel offsets (*multi-channel communication*). Hence, each cell in the TSCH two-dimensional slotframe is identified through a couple of information (*timeslot, channel*

offset). Finally, to contrast multi-path fading and interferences, TSCH relies on *channel hopping*. A pre-defined channel-hopping sequence is shared among all the nodes so that they can select a different operating frequency at each timeslot.

TSCH cells are allocated to nodes according to a communication schedule which specifies when a node can transmit/receive, and the frequency to be used. Cells may be either *dedicated* or *shared*. Dedicated cells are assigned to one specific node for data transmission and are guaranteed to be contention free. Instead, shared cells are allocated to all (or many) nodes and, hence, they are accessed on a contention basis, regulated by a TSCH CSMA algorithm [1].

The TSCH protocol does not specify how cells are allocated to nodes for communication. This is under the responsibility of the overlying 6TiSCH Operation sublayer (6top) that provides the abstraction of IP link over TSCH and manages the allocation of cells to nodes in such a way to meet the application requirements. At the higher layers, the 6LoWPAN adaptation protocol is used to fit IPv6 datagrams into TSCH packets, while the RPL routing protocol ensures multi-hop packet delivery.

2.1 6TiSCH Resource Management

The 6TiSCH architecture includes four different paradigms for building and maintaining the TSCH schedule, namely *static*, *centralized*, *distributed (*or *neighbor-to-neighbor)*, and *hop-by-hop scheduling* [30]. In static scheduling the schedule is static and either pre-configured or learned by a node at joining time. Typically, it is used during the network bootstrap or whenever a proper schedule is not available. Centralized scheduling relies on a central entity, the *Path Computation Element (PCE)*, responsible for: (i) collecting topology information and traffic requirements; (ii) computing the schedule; and (iii) disseminating it to all the nodes. In distributed scheduling, nodes agree on a common schedule using a neighbor-to-neighbor approach. They rely on a Scheduling Function (SF) to estimate the required number of cells, and use the 6P protocol to negotiate their allocation and deallocation with neighbors. Finally, hop-by-hop scheduling consists in reserving cells along a path for a particular traffic flow. In the following, we will refer to distributed scheduling based on neighbor-to-neighbor negotiation trough the 6P protocol.

2.2 6P Protocol

The 6P protocol defines the operations and messages to implement a 6P transaction, i.e., a complete negotiation between two neighboring nodes. Two different configurations are defined, namely, *2-step transaction*, based on a *Request* and a *Response* message, and *3-step transaction*, which includes an additional *Confirmation* message. Usually, the 2-step transaction is used (e.g. in [3, 25, 27]), as it ensures a lower overhead in terms of exchanged control messages. The Request message can be of different types, namely, ADD (to request a new allocation), DELETE (to remove an existing allocation), and CLEAR (to reset completely the current allocation). The Response message can be either a SUCCESS or an ERROR message, to notify a successful or failed transaction, respectively.

Fig. 2 illustrates the message exchange in a 2-step transaction. The left-hand side shows a successful transaction. A node requires the allocation of a specific cell, by sending an ADD message; its neighbor accepts the request and replies with a SUCCESS message. When a message gets lost or corrupted (right-hand side), a retransmission of the ADD message is performed after a timeout (6P Timeout). To keep track of the modifications to the schedule, 6P messages include a generation number, which is incremented every time the schedule changes. When a mismatch in the generation counter is detected (e.g., because a Response message got lost), the node replies with an ERROR message that forces the node that issued the 6P transaction to send a CLEAR message and reset the schedule. Then, the node has to start over the allocation process. The standard allows each node to execute only one 6P transaction at a time.

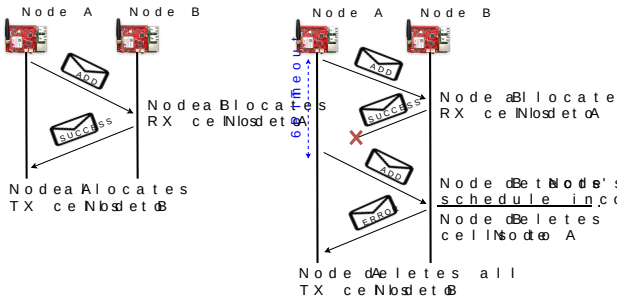


Fig. 2. 6P message exchange.

3 RELATED WORK

Since scheduling is a key problem in 6TiSCH networks, many previous papers have addressed this aspect. The proposed solutions can be broadly classified according to the four paradigms currently considered by the 6TiSCH WG [30], i.e., *static*, *centralized*, *distributed*, and *hop-by-hop* scheduling. However, *autonomous* and *hybrid* scheduling solutions also exist.

Static scheduling consists in a pre-established schedule shared by all nodes in the network, either pre-configured or learned by nodes at joining time. An example of static scheduling is the *minimal scheduling* defined in [34] and used at bootstrap (or when a proper schedule is not available). A variation of static scheduling is the *Adaptive Static Scheduling* presented in [14]. It consists in a static schedule with over-allocation and allows each pair of communicating nodes to activate only a subset of their assigned cells. Each node can activate/deactivate cells dynamically, depending on the experienced level of utilization, thus reducing the idle listening overhead of unused slots. Although both Adaptive Static Scheduling [14] and the proposed E-OTF algorithm increase/decrease the number of cells to use, depending on traffic conditions, the two solutions take a different approach in adapting to time-varying traffic conditions. E-OTF explicitly considers the number of retransmissions when computing the required number of cells, and includes a mechanism to react to congestion at nodes when the queue size exceeds a given threshold. Instead, the algorithm in [14] activates/deactivates at most one cell when the cell utilization is beyond/below a predefined threshold (cells are not released until the queue is completely empty). In this respect, Adaptive Static Scheduling [14] is more similar to MSF [4].

In *centralized scheduling* a central entity is responsible for collecting topology and traffic information, and computing the schedule [19, 20, 31]. Centralized algorithms are not suitable for large-scale and/or dynamic networks, as they require a high communication overhead and are unable to quickly adapt to time-varying traffic and network conditions. To overcome these issues, in AMUS [19] the central entity assigns additional cells to nodes with vulnerable links, to be actually allocated only if retransmissions are required. In addition, the schedule is computed by daisy-chaining cells from the leaves to the root, thus minimizing latency and avoiding collisions. In [31] the authors show how centralized scheduling and Software Defined Networking (SDN) can cooperate to allow dynamic operations on deterministic behaviors.

To overcome the limitations of centralized solutions, in *distributed scheduling* each node calculates the number of required cells, using a SF, and negotiates their allocation with neighbors through the 6P protocol. Many solutions for distributed scheduling are available in the literature. The algorithms presented in [2, 3, 16] take a distributed approach but rely on a coordinator node (e.g., the RPL root node). They share a common approach in which all the nodes send their traffic and topology information to the coordinator, which then triggers a distributed allocation procedure by

sending control messages. The latter include different information, depending on the specific policy used by each node to implement the local allocation process. Although these solutions leverage a distributed approach, a significant exchange of control messages is required for coordination. Hence, like centralized solutions, these algorithms are effective only when the network topology is (almost) static and traffic conditions are quite stable.

The *Minimal Scheduling Function* (MSF) [4] is currently under consideration by the 6TiSCH WG for standardization as the reference distributed SF, while previously the *On-The-Fly (OTF) Bandwidth Reservation algorithm* [27] [9] was considered. OTF takes a completely distributed approach where each node dynamically estimates the number of required cells, based on traffic conditions, and triggers 6P transactions for their allocation or deallocation. On the other side, MSF allocates two different kinds of cells, namely, *autonomous* and *negotiated* cells. Autonomous cells are intended to provide a basic amount of bandwidth for control messages and are allocated by nodes themselves using a hash function on node addresses (i.e., without any neighbor-to-neighbor negotiation). Instead, negotiated cells are allocated and deallocated dynamically, based on the level of utilization, through the 6P protocol.

The solutions proposed in [5, 24, 25] take a fully distributed approach, like OTF. However, they are designed with some specific goal in mind, e.g., low end-to-end latency [5], scalability [24], no overlapping allocations [25]. On the same note, the PID algorithm [8] aims at minimizing the amount of packets in the queue. The latter, however, takes an approach to queue management significantly different from that used in OTF and other distributed SFs (e.g., [5], [25] and [4]). Specifically, PID leverages the Proportional, Integral and Derivative (PID) control paradigm to derive the number of cells to allocate.

All the previous distributed SFs rely on the 6P protocol for cell negotiation. Instead, *autonomous scheduling* [10, 13, 17, 18, 22] exploits topology and local information to build the schedule autonomously, i.e., without any negotiation among nodes. To allocate cells, nodes use a hash function applied to the address of the same node and its neighbors. A survey of autonomous scheduling methods is available in [13]. Specifically, Orchestra [10] provides two types of scheduling, namely receiver-based and sender-based. In *receiver-based* scheduling, each node has only one fixed cell to *receive* (unicast) packets from any other node, while in *sender-based* scheduling each node has only one cell to *send* (unicast) packets to any other node. ALICE [22] extends this approach by replacing the *node-based* scheme used in Orchestra with a *link-based* scheme which allocates a unique cell for each unidirectional link. In addition, to minimize the impact of possible conflicts due to the hash function, the schedule is re-computed at each slotframe. To overcome the lack of flexibility in previous solutions, TESLA [18] allows to handle unpredictable changes in traffic and/or topology by extending the Orchestra receiver-based approach. Each node monitors its traffic load and changes the slotframe size according to its level of congestion. TESLA guarantees an autonomous adaptation to dynamic conditions at the cost of propagating the new slotframe sizes to neighbors. Similarly, in OST [17], nodes dynamically adapt the frequency of cells to time-varying traffic conditions. Moreover, they feature on-demand cell allocation to manage bursty/queued packets.

To complete our survey on 6TiSCH scheduling approaches, we need to mention *hop-by-hop scheduling* that consists in cell reservation at intermediate nodes along the path from a source to a destination performed by using an end-to-end signaling. The protocol for cell reservation is not yet defined, however, it is expected to be similar to reservation protocols currently used in the today Internet. For instance, in [23] the authors propose a subset of the RSVP-TE protocol.

Finally, *hybrid* solutions that combine different previous scheduling approaches also exist [12, 21]. In [21], the authors investigate the coexistence of applications with very different requirements (e.g., real-time and no-real time traffics) and propose a hybrid approach where dedicated and shared cells are used in the same schedule. On the same note, Elsts et al. [12] propose a hybrid approach to cope

with unpredictability of traffic. The proposed scheme relies on static scheduling to provide a basic amount of dedicated cells to nodes. In addition, over-allocation is used to cope with excess traffic due to variable data rate, node mobility, wireless link dynamics or route changes. Over-allocation leverages shared TSCH cells.

In this paper, we address the distributed mode of 6TiSCH and, hence, we mainly focus on distributed scheduling based on neighbor-to-neighbor negotiation. However, in our performance evaluation we also consider MSF [4] (the SF under consideration for standardization by the 6TiSCH WG), which includes both an autonomous component, inspired by the proposals in [10, 22], and a distributed component based on neighbor-to-neighbor negotiation. To the best of our knowledge, this is the first work in which MSF is extensively analyzed through both simulations and real experiments.

Most of the distributed scheduling solutions mentioned above have been evaluated in a number of studies [2, 3, 5, 8, 16, 25, 27]. However, for the analysis in [24, 27] the authors assume that 6P transactions are instantaneous and always successful, while in other studies large node buffers are considered [29]. In this paper, we consider a realistic wireless channel model and a limited buffer size similar to the one available on real IoT devices. In addition, we show analytically that, in a real environment, the duration of a 6P transaction may take up to several seconds to complete, and even fail. This has a significant impact on the performance of the SF, as confirmed by our experiments.

While many studies have analyzed the performance of the SF considered in isolation [5, 8, 25, 27], only few works have investigated the interplay of the SF with other IoT protocols. In [15, 33] the authors analyze the interplay between RPL and the 6TiSCH minimal scheduling [34], i.e., the static schedule used at bootstrap. However, the study only addresses the bootstrap phase and aims at investigating the impact of the minimal scheduling on the performance of RPL. In [28] the interplay between RPL and 6P protocol is investigated, however no specific SF is considered. In [22] the authors analyze the relationship between RPL and TSCH and exploit this relationship to define the autonomous schedule through ALICE. Unlike [22], in this paper we investigate the interplay between RPL and TSCH when a dynamically changing number of cells are negotiated through 6P. Such dynamics are not considered in [22], as ALICE adopts an autonomous scheduling that does not require negotiation. Specifically, we consider the effects of both 6P and RPL on the performance of the SF. To the best of our knowledge, this is the first work analyzing the distributed resource management mode of 6TiSCH in all its components.

This paper extends a previous conference paper by the same authors [29], where a preliminary analysis of OTF was carried out and E-OTF was originally proposed. In this paper, we consider MSF and PID, in addition to OTF. The analysis is no more limited to the SF, but considers all the components of 6TiSCH distributed scheduling (i.e., it includes an analytical model to analyze the performance of the 6P protocol). Finally, simulation results are validated by a set of experimental measurements in a real testbed.

4 ANALYSIS OF THE 6P PROTOCOL

Before analyzing the complete system by means of simulation and real experiments, we focus on the 6P protocol. The objective of this study is to highlight that the assumption of instantaneous and always successful 6P transactions [24, 27] is not realistic and can lead to biased results. To this end, we first derive an analytical model of the 6P protocol (Section 4.1) and, then, we present the results provided by the same model (Section 4.2). It is important to highlight that our goal is not to derive a precise analytical model of the 6P protocol. Instead, we aim at providing a model that could offer a good estimation of the average time required to complete a 6P transaction and the likelihood of its failure to demonstrate that, contrary to what assumed in many previous studies, 6P transactions may take a significant time to complete and even fail. Although in the design of the model we

favored tractability over fidelity, the accuracy of the obtained results is proved satisfactory by means of simulations.

4.1 Analytical Model

The analysis is based on a *Discrete Time Markov Chain (DTMC)* that models a 6P transaction between a generic node and one of its neighbors, i.e. the transmission of a Request message and the related Response message. Our objective is to model only the exchange of 6P messages and the required TSCH channel access procedure to estimate the delay for the completion of the transaction and the probability of its success/failure. Specifically, our goal is to highlight the following: (i) the time required to complete a 6P transaction is not negligible; (ii) 6P transactions can fail, especially when a lossy channel is considered. Considering that the objective is to gain an insight on the 6P protocol without external influences, the model does not consider a specific SF, neither it takes into account the dynamics of the routing protocol, whose instability can impact the performance of 6P transactions, as shown in [28]. Moreover, competing transactions issued at the same time by different nodes are not considered in the model, thus measuring the 6P transaction completion time and failure probability in a best case scenario.

In our model we assume that 6P messages are transmitted only on a set of n shared cells (6P cells), which are specifically reserved for 6P transactions. Such cells are assumed to be statically allocated by every node at network bootstrap and equally spaced within the slotframe. The assumption of equally spaced cells ensures that a node has the opportunity to transmit a 6P message every d seconds, which is the time between two consecutive *6P cells*. According to TSCH specifications, messages on shared cells are transmitted following the TSCH CSMA/CA algorithm, defined in the TSCH standard [1]. Every time a 6P message is ready for transmission, the message is transmitted on the next 6P cell. If the transmission is unsuccessful, a retransmission is issued according to a backoff procedure. Specifically, the sending node selects a random number of shared cells to wait before transmission. The selection is performed in the interval $[0, 2^{be} - 1]$, where be is the current backoff exponent. The latter is initially set to $minBackoff$ at the time of the first retransmission and increased for every retransmission to double the backoff window. The backoff exponent be is incremented until a maximum value $maxBackoff$. Specifically, after the $i-th$ retransmission, be is updated as follows: $be = min(maxBackoff, minBackoff + i)$. As result, the number of possible doublings for the backoff interval is $NB = maxBackoff - minBackoff$. In our analysis, for the

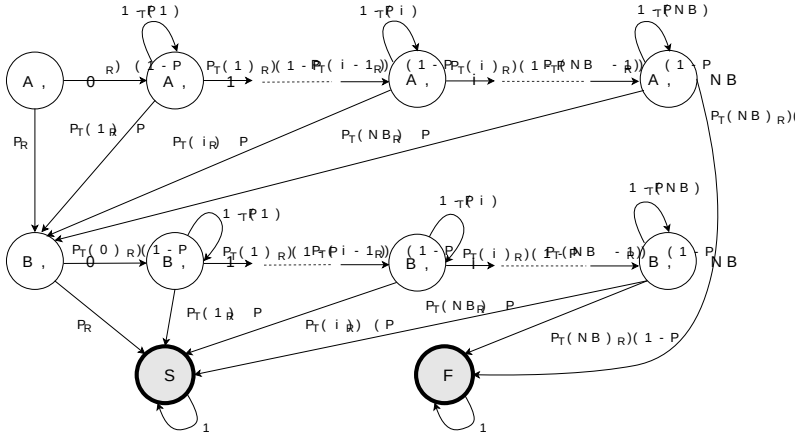


Fig. 3. Discrete Markov Chain model, graphical representation.

sake of simplicity, we assumed that NB is also equal to the maximum number of retransmissions that can be issued before canceling the transmission.

Figure 3 shows the graphical representation of the Discrete Markov Chain. Two absorbing states, S and F, represent the termination of the 6P transaction in a successful or unsuccessful manner, respectively. The other states $\{(A, i) \mid 0 \leq i \leq NB\}$ and $\{(B, i) \mid 0 \leq i \leq NB\}$, model the channel access operations performed on each shared cell for the transmission of a Request message and a Response message, respectively. Specifically, state $(A, 0)$ models the first transmission of a 6P Request message, while states $\{(A, i) \mid 1 \leq i \leq NB\}$ represent, each one, the $i-th$ retransmission of the same message. In particular, state (A, NB) represents the last retransmission, after which the Request message is dropped and, consequently, the 6P transaction is aborted. In a dual manner, state $(B, 0)$ models the first transmission of a 6P Response message, while states $\{(B, i) \mid 1 \leq i \leq NB\}$ represent, each one, the $i - th$ retransmission of the Response message, including the last retransmission modeled by (B, NB).

The transaction starts from state $(A, 0)$, corresponding to the first transmission of the Request message. Considering that the message is transmitted at the first available cell, the transaction leaves state $(A, 0)$ after the first transmission. Assuming that a message is successfully received with a fixed probability P_R, the transaction moves to state $(B, 0)$ with probability P_R, while it moves to $(A, 1)$ with probability $(1 - P_R)$, which represents the probability that the Request message is not successfully received. According to the TSCH CSMA/CA procedure, the retransmissions are performed following a backoff algorithm, in which a random time interval, depending on the number of previous unsuccessful retransmissions, is waited. In order to model the random waiting time before transmission on each backoff state, each $\{(A, i) \mid 1 \leq i \leq NB\}$ state is left with a certain probability $P_T(i)$, while, the transaction remains in the same state with a probability $1 - P_T(i)$. $P_T(i)$ is the transmission probability on a cell, when the backoff procedure is on the $i - th$ state, i.e. i previous unsuccessful transmissions have occurred. According to the TSCH CSMA/CA backoff procedure, on the $i - th$ state a cell is randomly selected for transmission in the interval $[0, E]$. E is the current backoff length set as $E = 2^{minBackoff+i-1} - 1$, where $minBackoff$ is the minimum backoff exponent and i is the current backoff exponent. Although during the execution of the backoff procedure, the probability of transmitting a message on a cell changes over time, for the sake of simplicity we set $1 - P_T(i)$ equal to the average transmission probability, which can be expressed as $\frac{E}{2}$[1] Consequently, the transmission probability $P_T(i)$ can be expressed as follows:

$$P_T(i) = \frac{2}{E} = \frac{2}{2^{minBackoff+i-1} - 1} \tag{1}$$

Any $\{(A, i) \mid 1 \leq i \leq NB\}$ state, instead, is left with $P_T(i)$ probability.

From any state $\{(A, i) \mid 1 \leq i \leq NB - 1\}$ the transaction moves to state $(A, i + 1)$ if the message is not correctly received. This transition occurs with probability $P_T(i)(1 - P_R)$, which represents the probability that the Request message is not correctly received, conditioned on the event of its transmission. Similarly, from state (A, NB) the transaction moves to the absorbing state F with the same probability. Since the maximum number of retransmissions has been reached, the 6P transaction is marked as failed. In case the message is successfully received, from any $\{(A, i) \mid 1 \leq i \leq NB\}$ state, the transaction moves to state $(B, 0)$. This transition occurs with probability $P_T(i)P_R$, which is the probability that the Request message is successfully received, conditioned on the event of its transmission.

[1] An accurate model of single backoff operations would have required a larger number of states in the Markov Chain, following the approach in [7]. The inaccuracy introduced by this simplifying assumption in the model is assessed afterwards by means of simulations.

$$
P = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
0 & (1-P_R) & 0 & 0 & \cdots & 0 & P_R & 0 & 0 & 0 & \cdots & 0 & 0 \\
\hline
0 & 1-P_T(1) & P_T(1)(1-P_R) & 0 & \cdots & 0 & P_T(1)P_R & 0 & 0 & 0 & \cdots & 0 & 0 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & 0 & 1-P_T(i) & P_T(i)(1-P_R) & \cdots & 0 & P_T(i)P_R & 0 & \cdots & \cdots & \cdots & 0 & 0 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & 0 & 0 & 0 & 0 & 1-P_T(NB) & P_T(NB)P_R & 0 & \cdots & \cdots & \cdots & 0 & P_T(NB)(1-P_R) \\
\hline
0 & 0 & 0 & 0 & \cdots & 0 & 0 & (1-P_R) & \cdots & \cdots & \cdots & P_R & 0 \\
\hline
0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 1-P_T(1) & P_T(1)(1-P_R) & \cdots & \cdots & P_T(1)P_R & 0 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 1-P_T(i) & P_T(i)(1-P_R) & \cdots & P_T(i)P_R & 0 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 & 0 & 1-P_T(NB) & P_T(NB)P_R & P_T(NB)(1-P_R) \\
\hline
0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & \ddots & 1 & 0 \\
\hline
0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 1 \\
\hline
\end{array}
$$

Fig. 4. Transition Probability Matrix.

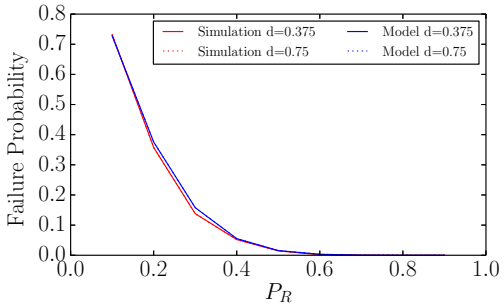


Fig. 5. Failure Probability of 6P transactions.

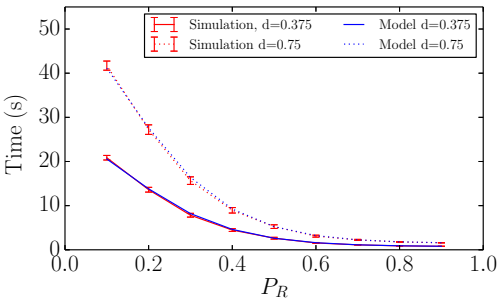


Fig. 6. Average completion time of 6P transactions.

The procedures performed for the transmission of the 6P Response are modeled through states $\{(B, i) \mid 0 \leq i \leq NB\}$. Their structure and transition probabilities are the same defined for modeling the transmission of the Request message, i.e. states $\{(A, i) \mid 0 \leq i \leq NB\}$. Specifically, from $(B, 0)$ the transaction moves with probability P_R to the absorbing state S, while it moves to state $(B, 1)$ with probability $(1 - P_R)$. In states $\{(B, i) \mid 1 \leq i \leq NB\}$ the transaction moves, with probability $P_T(i)P_R$, to the absorbing state S, while from any $\{(B, i) \mid 1 \leq i \leq NB - 1\}$ state the transaction moves with probability $P_T(i)(1 - P_R)$ to $(B, i + 1)$. Finally from state (B, NB) it moves to F, as the maximum number of retransmissions for the Response message is reached.

Figure 4 shows the Transition Probability Matrix associated to the Markov chain described so far. Exploiting the basic properties of the absorbing Markov chains, we evaluated the expected number of steps t before being absorbed in one of the two absorbing states and the absorbing probabilities g_F and g_S, on state F and S, respectively, starting from the initial state $(A, 0)$. In particular, t allows to calculate the average time to completion of a 6P transaction, while g_F and g_S allow to derive the expected failure and success probabilities, respectively.

4.2 Results

Considering the complexity of evaluating the Transaction Probability Matrix, shown in Figure 4, we solved the model numerically. In our evaluation, we considered an increasing range of P_R from 0.1 to 0.9 to consider different channel conditions. In addition, we set the following parameters: $minBackoff = 1$ and $maxBackoff = 7$. For the distance between two subsequent 6P cells d, we

consider two different values, $d = 0.375s$ and $d = 0.75s$, corresponding to a number of 6P cells equal to 4 and 2, respectively, considering a slotframe length of 101 cells. In order to validate the results obtained with the analytical model, a simple ad-hoc event simulator that simulates a node issuing a 6P transaction was written in Python. For each scenario 2000 independent replications were run. In Fig. 5 we show the average failure probability for each 6P transaction, while, in Fig. 6 we show the average time for the completion of a 6P transaction. The average 6P transaction failure and completion time obtained from simulations is reported along with the 95% confidence intervals. As can be seen, analytical and simulation results are almost overlapped.

The analytical (and simulation) results show that, when a lossy channel is considered (e.g., $P_R = 0.2$ or less), the failure probability can be very significant (i.e., larger than 40%). As can be seen the overall 6P completion time is in the order of seconds. Specifically it can range from a few seconds up to 20 seconds (with $d = 0.375s$) and 40 seconds (with $d = 0.75s$) when bad channel conditions occur. These results highlight that the assumption of instantaneous and always successful 6P transactions is unrealistic and can lead to inaccurate analysis. It is worth to mention that such results are obtained considering a best case scenario in which collisions from concurrent transmissions are not considered. In a real environment, concurrent transmissions can only increase the overall delay and the probability of failure, as shown in [28]. These results and the experience from [28], allowed us to draw a set of guidelines to set system parameters, such as the 6P timeout and the number of 6P cells in the slotframe. The values obtained, are adopted in the performance evaluation presented in the reminder of the paper.

5 SCHEDULING FUNCTIONS

In this Section we briefly describe the three SFs that will be considered in our performance evaluation in Section 6, namely, MSF, OTF and PID. For a complete description of the same SFs, the reader can refer to [4] [27] and [8], respectively.

5.1 Minimal Scheduling Function

MSF follows two different behaviors for two different phases in the network lifetime, i.e., the joining phase and the regular operation phase. During the joining phase, MSF implements the Minimal 6TiSCH Configuration [34]. Thus, when a node joins the TSCH network, its schedule has already a *minimal cell* installed. This is a shared cell for both TSCH and RPL broadcast control messages. In addition to the minimal cell, the node schedules two *autonomous cells*, which are allocated autonomously (i.e., without neighbor-to-neighbor negotiation) using a hash function to compute their timeslot and channel offset. There are two autonomous cells at each node, namely, an *Autonomous Rx Cell* and an *Autonomous Tx Cell*. The Autonomous Rx Cell is used by the node for receiving control messages and is allocated at joining time, while the Autonomous Tx Cell is allocated only after the node has selected its parent node (i.e., the neighbor selected by RPL for forwarding data towards the final destination) and it is a cell shared with all the other children of the same parent node. The Autonomous Rx Cell remains scheduled all the time, while the Autonomous Tx Cell changes if the parent node changes and is not permanently scheduled but added when there is a control message to send (and deleted just after).

The regular operation phase starts when the *minimal* and *autonomous* cells are scheduled and allows MSF to dynamically adapt to traffic requirements by allocating and deallocating *negotiated cells* through the 6P protocol. Initially, each node negotiates a single cell with its parent node. Then, it periodically checks the utilization of the negotiated cell(s) and increases/decreases their number in such a way to keep the utilization within two pre-defined thresholds. As shown in Algorithm 1, if the number of used cells is higher than the upper threshold, one additional cell is negotiated

ALGORITHM 1: MSF Algorithm (*NumCellsElapsed*, *NumCellsUsed*)

Input:

NumCellsElapsed = Number of elapsed negotiated cells
MAX_NUMCELLS = Max number of elapsed negotiated cells
NumCellsUsed = Number of negotiated cells used for transmission
LIM_NUMCELLSUSED_HIGH = Threshold to add a new negotiated cell
LIM_NUMCELLSUSED_LOW = Threshold to delete an unnecessary negotiated cell

Output:

ADD/DEL one negotiated cell

1 **if** *NumCellsElapsed* > *MAX_NUMCELLS* **then**
  if *NumCellsUsed* > *LIM_NUMCELLSUSED_HIGH* **then**
    trigger 6P to ADD one negotiated cell
2 **else if** *NumCellsUsed* < *LIM_NUMCELLSUSED_LOW* **then**
    trigger 6P to DEL one negotiated cell

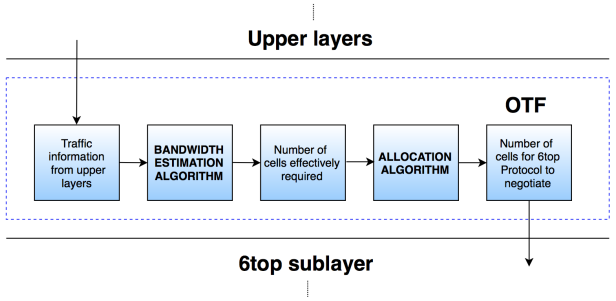


Fig. 7. Main components of the OTF Scheduling Function.

with the parent node. Similarly, when the cell utilization falls below the lower threshold, one cell is deallocated.

5.2 OTF Scheduling Function

As shown in Fig. 7, OTF [27] has a modular design and includes two main components, namely, a *Bandwidth Estimation Algorithm* and an *Allocation Algorithm*. The former is responsible for estimating the bandwidth required to transfer data to the parent node. To this end, it continuously monitor the data traffic towards the parent node. Then, the estimated bandwidth is passed, as an input, to the Allocation Algorithm that (i) computes the number of required cells, and (ii) starts a 6P transaction for negotiating their allocation with the parent node. In order to reduce the fluctuations of the bandwidth estimation, that would result in frequent addition/deletion of cells, the Allocation Algorithm leverages a hysteresis mechanism that usually results in over-provisioning.

The Allocation Algorithm, is described in Algorithm 2. The computation of the required number of cells is based on the following information. S_c: number of currently scheduled cells; R_c: estimated number of cells, provided by the Bandwidth Estimation Algorithm; T: hysteresis quantum. By comparing R_c with S_c, the Allocation Algorithm computes the number of cells to be added or deleted, also considering the hysteresis quantum T. As soon as R_c increases with respect to S_c, new cells are added (line 1). Instead, when R_c decreases, cells are deleted. The latter operation, however, is performed only if the difference between S_c and R_c is larger than the hysteresis quantum T (lines

ALGORITHM 2: OTF Allocation Algorithm (S_c, R_c, T)

Input:

S_c = Number of scheduled cells
R_c = Number of required cells
T = Hysteresis Quantum

Output:

ΔS = Number of cells to add/delete

1 **if** $R_c > S_c$ **then** $\Delta S = R_c - S_c + \lceil T/2 \rceil$ $\Rightarrow$ ADD CELLS
2 **else if** $R_c < S_c - T$ **then** $\Delta S = S_c - R_c - \lfloor T/2 \rfloor$ $\Rightarrow$ DELETE CELLS
3 **else** $\Delta S = 0$ $\Rightarrow$ DO NOTHING

2-3) to adopt a conservative policy. Whenever a new allocation is computed, a 6P negotiation is triggered.

5.3 PID-Based Scheduling Function

PID [8] estimates the number of cells to allocate by exploiting the well-known Proportional, Integral, and Derivative (PID) paradigm used in industrial control applications. Like OTF, PID is composed of two main modules, namely the *Timeslot Estimation Algorithm*, responsible for estimating the required number of cells, and the *Allocation Algorithm* that implements the control-loop feedback mechanism to actually allocate cells. The estimation performed by the Timeslot Estimation Algorithm is based on the queue occupancy and the current number of scheduled cells. At each execution, the algorithm computes the required number of cells, according to a parameter P_{tj}, which is the target amount of packets waiting in the transmission queue for the parent node j. Specifically, the number of cells to be added or released, C_{sj}, is computed as follows:

$$C_{sj} = K_p * e_{tj} + K_i \sum_{0}^{n-1} e_{tnj} * \delta_{SF} + K_d * \frac{e_{tj} - e_{t-1j}}{\delta_{SF}} \tag{2}$$

where e_{tj} is the error evaluated after each allocation, and it is defined as follows:

$$e_{tj} = P_{cj} - C_{cj} - P_{tj}. \tag{3}$$

In Equation 3, P_{cj} is the number of packets currently in the queue, while C_{cj} is the current number of scheduled cells.

Equation 2 can be decomposed in three terms, namely the proportional, integral and derivative terms. The integral term $\sum_0^{n-1} e_{tnj}$ is the integral of the errors of the last n slotframes, while δ_{SF} represents the duration of the slotframe. The derivative term $\frac{e_{tj}-e_{t-1j}}{\delta_{SF}}$ predicts future values of the error by considering the last two ones. The proportional term $K_p * e_{tj}$ reacts proportionally to the actual error to avoid unnecessary exchange of 6P messages. Finally, the three parameters K_p, K_i and K_d, all ranging in [0,1], are used to weight each term of the equation. Specifically, K_p weights the actual error and should be close to 1, K_i weights the integral term and, in order to stabilize the controller, should be significantly smaller than K_p, while K_d weights the future behavior of the systems and contrasts its inertia.

6 PERFORMANCE EVALUATION OF THE SCHEDULING FUNCTIONS

In this section, we present the results of the performance evaluation of the three SFs described in Section 5. First, we introduce our simulation methodology and, then, we discuss the obtained results.

Table 4. **Simulation setting and parameters**

Simulation setting and parameters	
TSCH slotframe length	**101**
TSCH timeslot duration	**15ms**
TSCH number of channels	**16**
Buffer size	**8 packets**
RPL redundancy threshold	**10**
RPL minimum interval	**128ms**
RPL maximum doublings	**20**
RPL Objective Function	**MRHOF - ETX**
MACMinBE - MACMaxBE	**1 - 7**
6P Timeout	**50s**
6P Cells	**4**
Traffic Rate (packets/slotframe)	**2, 1, 1/2, 1/3, 1/4, 1/6, 1/8, 1/10**

Table 5. **SF parameters**

SF parameters	
MSF MAX_NUMCELLS	**64**
MSF LIM_NUMCELLSUSED_HIGH	75% MAX_NUMCELLS **48**
MSF LIM_NUMCELLSUSED_LOW	25% MAX_NUMCELLS **16**
OTF T (hysteresis quantum)	**2, 4, 8**
PID $K_p - K_i$	**0.7 - 0.075**
PID K_d - P_{tj}	**0 - 0**
PID n	**4**

6.1 Simulation Methodology

In order to assess MSF, OTF and PID under realistic conditions, we implemented them in ContikiOS [2], an operating system for sensor nodes. ContikiOS already implements the basic functionalities of the TSCH protocol [11]. In addition, it includes an implementation of the whole 6TiSCH protocol stack, namely the 6LoWPAN header compression protocol and the RPL protocol. In order to carry out an extensive performance evaluation, we used the Cooja simulator [26], a network simulator available as part of the Contiki distribution. In our simulations, Cooja has been configured to simulate a network of Cooja motes, i.e., generic sensor nodes equipped with an IEEE 802.15.4 radio.

We considered two different simulation scenarios. A very simple scenario, to analyze the behavior of each SF in a controlled environment, and a second, more complex scenario, to assess the performance of the considered SFs in a more realistic environment. The simple scenario consists of a network with 7 nodes arranged in a *chain topology*, as shown in Fig. 8. The distance between neighboring nodes is 33m. The second scenario considers a network with a regular *grid topology*, namely a 5x5 grid with 25 nodes, as shown in Fig. 9. The distance between neighboring nodes (located on the same row/column of the grid) is set again to 33m. In both scenarios, to simulate a realistic wireless channel, we adopted the *Multi-path Ray-tracer Medium (MRM)* model [26]. MRM is a realistic propagation model that implements ray-tracing techniques with various propagation effects, e.g., multi-path, refraction, diffraction, etc. Each link in the network has an associated average Packet Delivery Probability (PDP), which however changes over time due to the propagation effects and concurrent transmissions. In our simulations the MRM channel parameters are set in such a way to result in a PDP of 100% at 33m distance when using the maximum transmission power. The PDP of the other links is reported in both Fig. 8 and Fig. 9 for the chain and grid scenarios, respectively.

[2] Contiki OS, http://contiki-os.org/

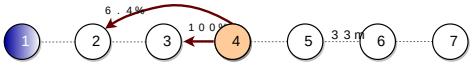


Fig. 8. Chain topology and Packet Delivery Probability.

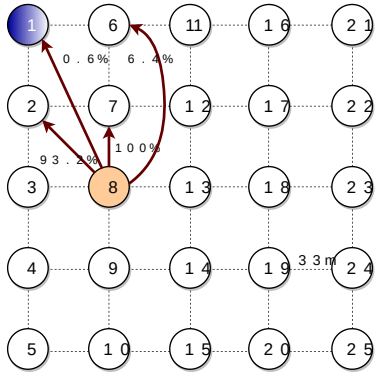


Fig. 9. Grid topology and Packet Delivery Probability.

In order to implement multi-hop communication, the RPL routing protocol is used in both scenarios. The 6P configuration parameters, i.e., *6P Timeout* (the timeout for each transaction) and *6P Cells* (the number of cells, in the TSCH slotframe, devoted to 6P messages) are reported in Tab. 4, as well as the parameter setting for TSCH and RPL. The values of *6P Timeout* and *6P Cells* are set according to the guidelines provided by the analytical results obtained in Section 4, and the experience gained from the study in [28]. In both the considered scenarios, node 1 is configured to behave as both data sink and RPL root. Non-root nodes are programmed to generate a periodic data traffic directed to the sink. Specifically, 30-byte packets are periodically generated, with different packet generation periods (i.e., *Traffic Rates*) as shown in Tab. 4. Each simulation is run for 60 minutes. Every node is configured to start the generation of packets 10 minutes after the start-up time. This is required to ensure the proper stabilization of RPL and TSCH initial schedule. To obtain statistically sound results, we performed 30 independent replicas of each run, for all the considered configurations. The simulation results presented below are averaged over all the replicas. We also derive confidence intervals using the independent replications method and 95% confidence level.

We performed a preliminary set of experiments with different values of P_{tj}, K_p, K_i and K_d, to find the optimal parameter setting for PID. For the sake of brevity, we do not show the results. However, our preliminary analysis confirmed the parameter setting used by the authors of PID in [8]. All the values adopted for PID in our simulations are summarized in Tab. 5. Regarding OTF, we varied the *hysteresis quantum* T. The range of values adopted are summarized in Tab. 5. For MSF we considered two different configurations that only differ in the number of cells allocated for control traffic (i.e., 6P messages, RPL messages, TSCH Enhanced Beacons) during the joining phase. Specifically, the "MSF std" configuration uses the number of cells suggested in the specifications [4] (i.e., one minimal cell shared by all the nodes and two autonomous cells for each node), while the "MSF" configuration uses the same number of cells as OTF and PID (i.e., 4 shared cells for 6P transactions and 4 shared cells TSCH/RPL messages). The latter configuration was considered to make the comparison fair.

In our analysis we mainly considered the following performance metrics:

- *End-to-end reliability*, defined as the ratio between the number of packets received by the sink and the total number of packets sent by all the nodes.
- *End-to-end latency*, defined as the time interval between the generation of a packet and its (correct) reception at the sink.

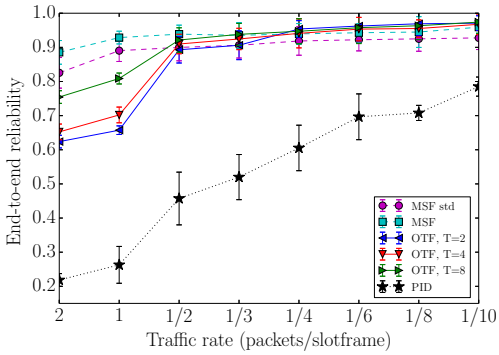


Fig. 10. End-to-end reliability. Chain topology.

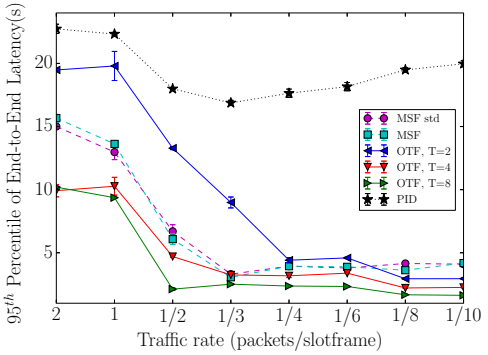


Fig. 11. 95^{th} Percentile of the end-to-end latency. Chain topology.

However, to better understand the behavior of the considered SFs, in some cases we also considered additional metrics, such as number of 6P transactions issued, number of scheduled cells, and queue size at nodes.

6.2 Simulation Results

6.2.1 Chain topology. We first discuss the results obtained in the chain topology. Fig. 10 shows the end-to-end reliability provided by MSF (in both the considered configurations), OTF and PID, for different traffic rates. For OTF, different values of T are also considered. Dashed lines refer to MSF, solid lines correspond to OTF, and the dotted line is for PID. PID exhibits a very low reliability, i.e., in the range between 22% and 79%, significantly lower than that provided by MSF (83%-96%) and OTF (63%-97%). The behavior of OTF does not depend significantly on the value of T, unless for the highest considered traffic rates (1-2 packets/slotframe), where more over-provisioned cells help in increasing the end-to-end reliability. MSF achieves better end-to-end reliability, compared with PID and OTF, especially at high traffic rates (i.e., 1-2 packets/slotframe).

The 95^{th} percentile of the end-to-end latency is shown in Fig. 11. PID always exhibits the highest latency, while the performance of OTF strongly depends on the value of the hysteresis quantum. Specifically, when a low T value is considered (e.g., $T = 2$), OTF results in a high latency, while the latter decreases significantly when larger T values are used. This can be easily understood if we consider that a larger T value implies a larger number of over-provisioned cells and, hence, a lower waiting delay experienced by packets. Both MSF configurations result in similar end-to-end latency, with values in between the results obtained with OTF when $T = 2$ and $T = 4$, respectively.

In MSF the degree of over-provisioning can be controlled by setting a proper value to the thresholds. In our experiments, we also analyzed the impact of the upper and lower threshold on the performance of the SF. Specifically, we performed a set experiments where we fixed the value of the lower threshold (25%) and varied the upper threshold (60%, 75%, 90%). Similarly, in a second set of experiments, we varied the lower threshold (15%, 25%, 35%), while keeping the higher threshold constant (75%). In both set of experiments, we did not observe significant variations in the end-to-end reliability and latency. The results are not shown in Fig. 10 and 11 (as well as in subsequent sections) to make the plots more readable.

To get an insight on the behavior of the SFs and understand the reasons for the low performance of PID, we focus on a single experiment. In Fig. 12, we show, for all the considered SFs, the total

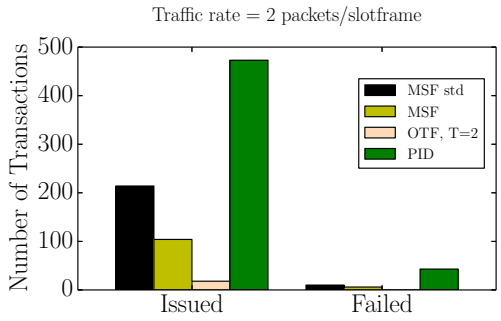
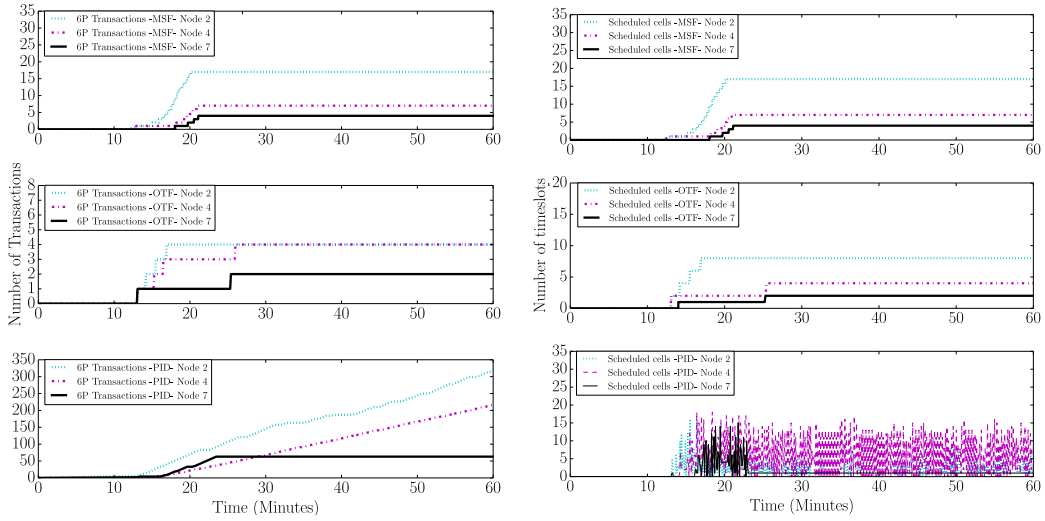


Fig. 12. Overall number of 6P transactions.

number of 6P transactions issued during the experiment by all the nodes in the network and the fraction of them resulting in a failure (the traffic rate is 2 packets/slotframe). OTF and PID generate the lowest and highest number of 6P transactions, respectively, while MSF is in between. In particular, MSF issues much more transactions in the standard configuration (i.e., with only 2 cells available for 6P transactions). This is because, using a lower number of cells for transmitting 6P transactions results in a higher collision probability and, hence, increased time to complete and higher failure rate.

To explain the difference in the number of 6P transactions issued by each SF, we compare the number of 6P transactions issued over time with the number of scheduled cells, shown in Fig. 13a and Fig. 13b respectively. For the sake of readability, we show the values for only three representative nodes (namely, nodes 2, 4 and 7), i.e., a node close to the sink, a node in the middle of the chain, and a node in the end of the chain. The values obtained with the "MSF std" are omitted for the sake of brevity. Fig. 13a (top) shows that with MSF, node 2 and 4 issue a certain number of 6P transactions that starts to increase linearly when the traffic of all the others nodes starts to circulate



(a) 6P Transactions MSF(top), OTF(mid), PID(bottom). (b) Scheduled cells MSF(top), OTF(mid), PID(bottom).

Fig. 13. 6P Protocol dynamics. Traffic rate: 2 packets/slotframe, for OTF T=2.

in the network. The value increases until it reaches the proper trade-off between the number of scheduled cells and their utilization (i.e., up to 17 for node 2 and up to 7 for node 4), as shown in Fig. 13b (top). Node 7 issues only 4 6P transactions in order to fulfill its traffic requirements. Fig. 13a (mid) presents the behavior of OTF when $T = 2$. Initially, node 2 and 4 issue one 6P transaction each, and allocate two cells each, as shown in Fig. 13b (mid). Then, all the other nodes start sending their data and, hence, node 2 and node 4 need to issue another 6P transaction each, to schedule two extra cells. These cells allow to forward the data generated by nodes far away from the sink. Node 7 issues two 6P transactions to schedule two cells for the transmission of its local traffic. As a global result, the overall number of 6P transactions issued by OTF is low and, consequently, they all succeed (as shown in Fig. 12). It may be worthwhile reminding that both MSF and OTF take a conservative approach and deletes cells only when they are not used for several consecutive slotframes (which is not the case in these simulation runs). By comparing Fig. 13b (top) and (mid) it emerges that OTF allocates a lower number of cells, with respect to MSF. This explains the lower end-to-end reliability provided by OTF, especially at high traffic rates (Fig. 10). PID exhibits a completely different behavior with respect to MSF and OTF, as shown in Fig. 13a (bottom). Nodes 2 and 4 issue a very high number of 6P transactions over time, changing the schedule very often. This is because PID computes the number of required cells with the goal of having no packets in the queue. Consequently, at each execution, it acquires the number of cells required to transmit *all* the packets in the queue, or it releases the unnecessary cells if the queue is (almost) empty. Therefore, the schedule is strictly dependent on the number of packets in the queue, which changes frequently. This appears very clearly in Fig. 13b (bottom). A high number of cells is required by node 2 and node 4, thus resulting in high peaks in the number of scheduled cells. Such peaks, are less pronounced for node 4 after the initial phase but remain constant for node 2, as it receives traffic from many descendant nodes. Overall, such a large number of 6P transactions increases the probability of failure, due to collisions, as confirmed in Fig. 12. We remember here that 6P messages are sent in shared cells.

The performance evaluation of PID, OTF and MSF in this simple scenario highlights that PID, in its current definition [8], is not suitable for a real deployment. It exhibits an unstable behavior due to the absence of a hysteresis mechanism to reduce the number of changes over time. This causes a large number of 6P transactions that can fail, due to collisions or buffer overflow, thus leading to delayed allocation of cells or inconsistencies in the schedule.

6.2.2 Grid topology. To evaluate the performance of MSF, OTF and PID, in a more complex scenario we also considered a 5x5 grid topology. Fig. 14 shows the end-to-end reliability provided by the three SFs in this scenario, for different traffic rates. The PID algorithm exhibits a very low reliability (in the order of 20%), while the performance of OTF is similar to the one observed in the previous scenario (the reliability values are between 58% and 99%). As far as MSF, we observe a greater gap between the two considered configurations in this more complex scenario as in a network of 25 nodes the negative impact of using less 6P cells is more apparent. In "MSF std" [4] nodes have only two cells dedicated to 6P transactions, one to issue a transaction to the parent node and one to reply to transactions from child nodes. Hence, a lot of 6P transactions can fail, or be delayed significantly. This does not allow schedule modifications to be executed rapidly, thus explaining the low performance of "MSF std". Instead, MSF with 4 cells for 6P transactions, achieves similar, or even better performance, in terms of end-to-end reliability, with respect to OTF with $T = 8$.

Looking at latency, Fig. 15 confirms that PID exhibits very low performance in this scenario, as the 95^{th} percentile of the end-to-end latency ranges between 15 and 22 seconds. With OTF, the end-to-end latency strongly depends on the hysteresis quantum, as the number of over-provisioned cells increases with T. As expected, latency increases with the traffic rate. When using MSF, latency

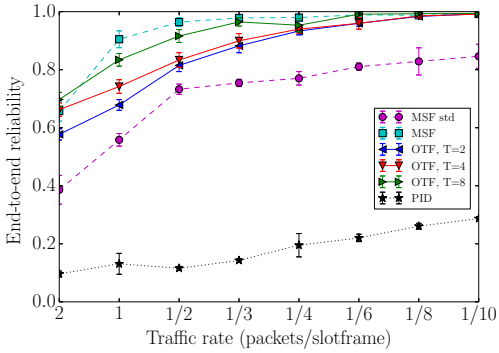


Fig. 14. End-to-end reliability. Grid topology.

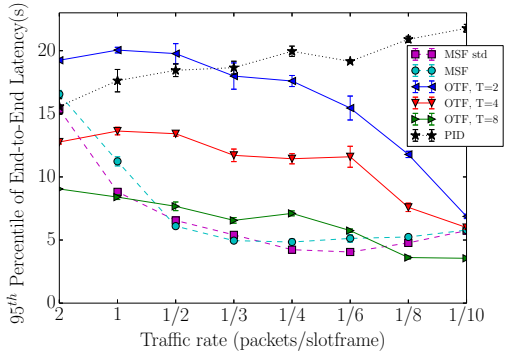


Fig. 15. 95^{th} Percentile of the end-to-end latency. Grid topology.

is high at high traffic rates (i.e., 1-2 packets/slotframe), while it quickly decreases as the traffic rate goes down. At the lowest traffic rates (i.e., 1 packet every 8 or 10 slotframe) a slight increase in the latency can be observed, as it occurred in the chain topology.

The results above highlight that the difference in performance, between MSF, OTF and PID, is even larger in this grid scenario, due to the larger number of nodes generating packets. This confirms that PID is not suitable for networks with traffic fluctuations caused by multiple traffic sources. Even though OTF and MSF exhibit better performance than PID, when the traffic rate is high (i.e., 1-2 packets/slotframe) their reliability is below 70% and the 95^{th} percentile of latency is between 15 and 20 seconds. These results may not be suitable for some (industrial) applications.

In OTF, the only parameter that can be exploited to increase the performance is the hysteresis quantum T. In principle, by increasing the number of over-provisioned cells it is possible to reduce latency and increase reliability, at the cost of additional bandwidth consumption. However, the above results show that, even with large T values, OTF may not be able to provide a high reliability and a low end-to-end latency, as desired. Similarly, for MSF we observed that using different values of the two thresholds does not result in a significant improvement (the results are not shown for brevity).

In order to investigate the reasons for this behavior, we measured the queue size at each node over time (i.e., the current number of packets enqueued for transmission on a node). Fig. 16 shows the queue size at node 7, both with MSF and OTF, in a specific experiment (for brevity, a single traffic rate is considered, however the behavior is similar for other traffic rates). We focus on node 7 because it is very close to the sink and, in addition, it is responsible for relaying traffic from several other descendant nodes. We can observe in Fig. 16 the two different behavior for MSF, (dotted line) and OTF (solid line). With OTF, node 7 starts to enqueue packets just after the traffic begins to circulate in the network and this is because the node has to negotiate the initial set of cells with its parent. As this negotiation terminates successfully, the node can start transmitting its packets and, consequently, the queue level decreases. After few minutes, however, the queue starts increasing again. We found that that this is due to a change in the parent node selected by RPL, which has found a potentially more convenient neighbor for data forwarding. The selection of a new parent node causes the deallocation of all the cells and a new negotiation with the new parent. During this period, the node has to buffer all its data packets. However, after the cells allocation towards the new parent has been completed, the queue level does not decrease, as

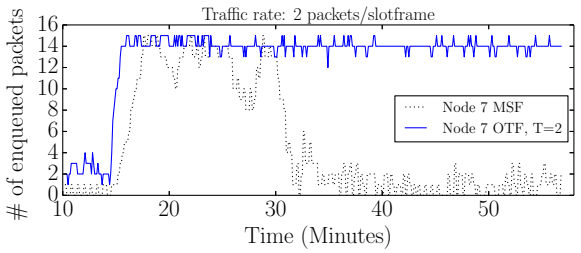


Fig. 16. Node 7 queue occupancy for MSF (dotted line) and OTF (solid line) with $T = 2$.

expected. Instead, it saturates the buffer. This happens because the link to new parent node is very lossy and, consequently, the successful transfer of packets requires multiple retransmissions. OTF does not consider, in its Bandwidth Estimation Algorithm, the number of expected retransmissions required for communicating with the parent node. Consequently, since additional retransmissions are required, the time for successful transmission of a packet increases, causing the number of packets in the queue to grow up. The behavior of MSF, instead, is different. Initially, as above, node 7 is able to forward its data, which results in a very low queue size. At a certain point, also in this case, a change in the parent node occurs, causing the queue occupancy to grow up until the maximum queue size is reached, as occurred with OTF. After some minutes, however, the node recovers from congestion. This so long time is due to the fact that MSF allocates just one more cell at a time by monitoring the slot utilization, which in this case is 100%. At around minutes 21 and 28, a neighbor selects node 7 as parent node. This increases the amount of traffic that node 7 has to handle, thus causing the number of the enqueued packets to increase again, waiting for the completion of 6P transactions to have the necessary amount of scheduled cells. After minute 33, node 7 has acquired the proper amount of cells to deliver its packets, consequently the amount of enqueued packet reduces and afterwards remains stable to 4, on average. By comparing the behavior of MSF and OTF, we can highlight that OTF is not capable of reacting to congestion at all. Instead, MSF is capable of reacting, however, the time required to detect and recover from congestion may be considerable, e.g., in the order of minutes, or tens of minutes.

As also highlighted in [28], the dynamism of the RPL protocol may result in frequent parent changes. In addition, the link-quality assessment mechanism used by RPL, in its default configuration, can lead to select a parent node characterized by a poor link quality. Both these factors can dramatically increase the queue size, resulting in large waiting delays, and possibly, packet dropping.

A more detailed analysis of our simulation traces also highlighted other *disruptive events*, which can force the node to change its schedule, deallocating all its cells at once. This leads to a significant increase of the queue, thus resulting in high waiting delays and packet dropping. Specifically, in addition to parent changes, we also observed failures of 6P transactions and TSCH de-synchronizations. The failure of a 6P transaction occurs whenever a 6P message gets corrupted or lost. This generates an inconsistency in the 6TiSCH schedule between the two neighbors. According to the 6P specifications, every time an inconsistency is discovered, the involved nodes must reset their schedule. This results in the deallocation of all the cells and, then, in a new negotiation phase to re-allocate the required number of cells. During this process, which may take even many seconds (as we showed in Section 4), the node cannot send or receive any packet, forcing data to be buffered. A TSCH de-synchronization, instead, occurs every time a node is not perfectly synchronized with its current parent. TSCH requires a rigorous synchronization among nodes to ensure that the sender and the receiver can correctly communicate on the same cell. If the de-synchronization of two

communicating nodes increases over a safe level, due to the skew of their internal clocks, the node is forced to disconnect from the TSCH network and start a new association process. Similarly to the previous case, the node must release all its cells and negotiate a new allocation from scratch after the new association.

7 ENHANCED-OTF

From the previous analysis it clearly emerges that both MSF and OTF exhibit a number of shortcomings. They are mainly related with RPL and 6P dynamics, which can significantly affect their performance. This is exacerbated by the fact that both the algorithms do not take into account the amount of data waiting for transmission in the computation of the required number of cells. This can lead to underestimate the number of cells to allocate during congestion. Specifically, MSF considers only the current cell utilization to establish if a cell should be added or deleted, whereas OTF makes an estimation of the bandwidth without considering the amount of enqueued packets. In addition to this, both MSF and OTF do not take in consideration information related with the quality of the link towards the parent node, e.g., the number of transmissions required to correctly deliver a packet.

In order to overcome these shortcomings, we propose a modified version of OTF, named *Enhanced-OTF* (E-OTF). We decided to build on OTF, instead of MSF, for the following reasons: (i) OTF embeds the T parameter that can be used to tune its behavior and, in particular, to select a proper level of over-provisioning that can meet the application requirements; (ii) OTF allocates cells based on the estimated bandwidth requirements, thus aiming, indirectly, at satisfying application requirements. MSF, on the other hand, allocates cells based on the current utilization, thus focusing on ensuring an efficient usage of resources, rather than satisfying application requirements. From our perspective, the OTF approach is more suited to ensure reliable and timed communication in critical applications for industrial environments. Hence, we decided to adopt OTF as starting point for our proposal.

In E-OTF the Allocation Algorithm of OTF is modified in such a way to: (i) explicitly consider the link quality in the computation of the required number of cells and (ii) include a mechanism to timely react when the queue size increases over a certain threshold. The resulting E-OTF algorithm is illustrated in Algorithm. 3. The differences with the OTF are emphasized.

First, the Bandwidth Estimation Algorithm is modified by introducing the *Expected Transmission Count (ETX)* [6] in the computation of the required number of cells (R'_c). Since ETX measures the average number of transmissions required to successfully transmit a packet on a link, the ETX value is used as a correction factor in the calculation of the required number of cells R'_c, i.e., $R'_c = R_c * ETX$ (line 1). This modification allows nodes with poor link quality towards their parent node to allocate more cells.

Second, a *bonus mechanism* is introduced to allow nodes to rapidly recover from congestion. To this aim, the computation of the number of cells is modified as follow. When the queue level exceeds a certain *congestion threshold* (β), the node enters a congestion state, where, at each execution of the algorithm, an additional number of cells B is acquired. This *congestion bonus* is used to over-provision the requesting node so as to allow it to transmit the buffered packets more rapidly. This modification to the original algorithm is required as the Bandwidth Estimation Algorithm does not take into consideration the amount of buffered data. When the amount of packets in the queue drops below the congestion threshold β, the node has to release the over-provisioned cells. Applying the default OTF policy, however, would result in releasing all the cells at once. In fact, R'_c, calculated according to the estimated bandwidth, would be significantly lower than the current S_c value that includes the bonus. In order to avoid this behavior, that might lead to congestion again, the node is forced to release the acquired bonus in a gradual manner. To this

ALGORITHM 3: Enhanced-OTF Allocation Algorithm ($S_c, R_c, T, B, Q, U, ETX, \beta, \alpha$)

Input:

S_c = Number of scheduled cells
R_c = Number of required cells
T = Hysteresis Quantum
B = Congestion Bonus (CB)
Q = Average Queue Occupancy
U = Average Cell Utilization
ETX = Estimated Link Transmissions

Output:

ΔS = Number of cells to add/delete

1 $R'_c = R_c * ETX$
2 **if** $Q > \beta$ **then** $\Delta S = B$ $\Rightarrow$ *ADD CB*
3 **if** $R'_c > S_c$ **then** $\Delta S+ = R'_c - S_c + \lceil T/2 \rceil$ $\Rightarrow$ ADD CELLS
4 **else if** $R'_c < S_c - T$ **then** $\Rightarrow$ DELETE CELLS
5 **if** $U > \alpha$ **then** $\Delta S = B$ $\Rightarrow$ *REMOVE CB*
6 **else** $\Delta S = S_c - R'_c - \lfloor T/2 \rfloor$
7 **else** $\Delta S = 0$ $\Rightarrow$ DO NOTHING

aim, the node periodically monitors the *cell utilization* (U) [3]. If the cell utilization is higher than a predefined threshold (i.e., $U > \alpha$), only the congestion bonus is released (line 7-8). Otherwise, when the utilization is very low ($U \leqslant \alpha$), the regular OTF policy is used, i.e., *all* the unnecessary cells are released at once (lines 9-10).

8 PERFORMANCE EVALUATION OF E-OTF

In this section, we assess the effectiveness of the proposed E-OTF algorithm against MSF and OTF (we omit to consider also PID and "MSF std" for the sake of brevity and, above all, due to their low performance). To this end, we performed an additional set of experiments considering the 5x5 grid scenario introduced in Section 6.1, and using the same parameter values reported in Tab. 4. Since E-OTF has three additional parameters, with respect to OTF (i.e., α, β, and B), we ran a set of preliminary experiments for tuning their values. Specifically, we selected parameter values that provided the best performance, in terms of end-to-end reliability and latency. Following this approach, we set the E-OTF parameters as follows: $\beta = 20\%$, $\alpha = 20\%$, $B = T$ (i.e., B is set equal to the value of hysteresis quantum). We used these values in all the experiments described below. In the following experiments, we considered the same performance metrics defined in Section 6.1. In addition, two other metrics were also considered:

- *Cell Utilization*, defined as the ratio between the number of cells used by a node and the number of cells allocated to it; it provides a measurement of the efficiency of the allocation policy.
- *Duty Cycle*, defined as the ratio between the number of cells in which the node is active and the size of the slotframe; it offers an indirect measurement of the energy consumption of a node.

Fig. 17 compares MSF, OTF and E-OTF, in terms of end-to-end reliability for different traffic rates (for OTF and E-OTF we also consider different T values). As expected, E-OTF significantly outperforms OTF. Specifically, E-OTF provides a reliability above 95% for *all* the considered traffic

[3] defined as the ratio between the number of cells actually used and the overall number of allocated cells

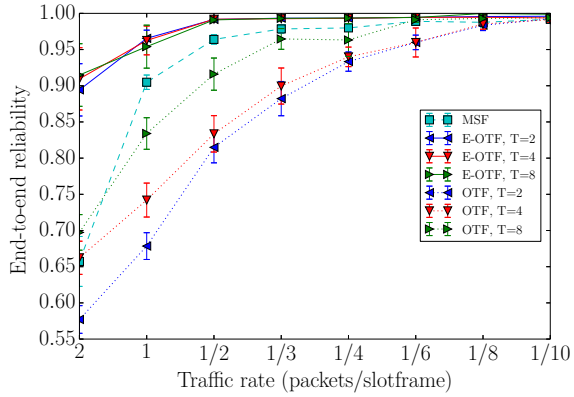


Fig. 17. End-to-end reliability in the grid topology.

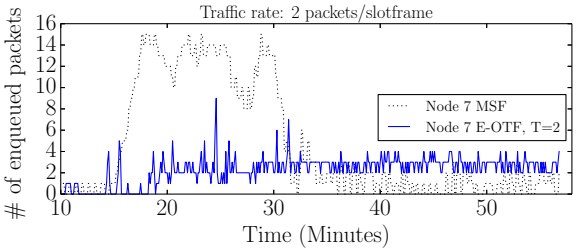


Fig. 18. Node 7 queue occupancy for MSF (dotted line) and E-OTF (solid line) with $T = 2$.

rates, except for the highest considered one, for which the end-to-end reliability is 89%. In addition, the difference in performance, with respect to OTF, increases as the traffic rate grows up, showing that E-OTF is able to handle high traffic conditions better than OTF. This is due to the modifications introduced in the estimate of the required number of cells, which allows to manage congestion in a rapid way, thus avoiding situations where packets are dropped, due to buffer overflow. The results in Fig. 17 confirm that estimating the required number of cells only based on the current traffic conditions, and without considering the buffered packets, is not an effective approach in a real environment. Moreover, E-OTF performs better than MSF as well, in all the considered scenarios, especially at high traffic rates (89% vs. 66% with 2 packets/slotframe). Again, this is due to the mechanism used by E-OTF to manage congestions (see below).

Tab. 6 compares the 95^{th} percentile of the end-to-end latency for different traffic rates. It emerges that E-OTF reduces latency by approximately 70-80%, with respect to OTF, especially at high traffic rates. With respect to MSF, the reduction in latency is slightly lower, but still very considerable, especially when considering E-OTF with $T = 4$ or $T = 8$. This reduction in latency is due to the ability of E-OTF to quickly react to congestion. This allows to rapidly reduce the queue at the node and, consequently, the waiting time experienced by packets. Although E-OTF achieves a dramatic reduction in latency, nevertheless, values remain in the order of several seconds. This is due to disruptive events that, of course, still occur, as they are caused by other protocols (i.e., RPL, 6P, and TSCH). However, their negative effects are strongly mitigated when using E-OTF and, hence, the reduction in end-to-end latency. Temporary increases in the queue level may still occur, however they are typically of short duration. To confirm this statement, we measured the overall number of

Table 6. 95^{th} Percentile end-to-end Latency(s).

T (Hysteresis Quantum)		2	4	8
packets/slotframe	MSF	16.6 ± 0.3		
	OTF	19.2± 0.1	12.8± 0.1	9± 0.02
2	E-OTF	2.9± 0.1	2.8± 0.1	1.9± 0.1
	Latency Reduction - MSF	**83%**	**83%**	**89%**
	Latency Reduction - OTF	**85%**	**78%**	**79%**
	MSF	11.2 ± 0.4		
	OTF	20.1 ± 0.2	13.6 ± 0.3	8.4 ± 0.2
1	E-OTF	4.3 ± 0.1	3.2 ± 0.1	2.1 ± 0.1
	Latency Reduction - MSF	**62%**	**71%**	**81%**
	Latency Reduction - OTF	**79%**	**77%**	**75%**
	MSF	6.1 ± 0.2		
	OTF	19.8 ± 0.8	13.4 ± 0.2	7.7 ± 0.3
1/2	E-OTF	3.6 ± 0.02	3.0 ± 0.03	2.5 ± 0.1
	Latency Reduction - MSF	**41%**	**51%**	**59%**
	Latency Reduction - OTF	**82%**	**78%**	**68%**
	MSF	4.9 ± 0.1		
	OTF	18.0 ± 1.0	11.7 ± 0.5	6.6 ± 0.2
1/3	E-OTF	4.1 ± 0.02	3.7 ± 0.02	2.9 ± 0.1
	Latency Reduction - MSF	**16%**	**24%**	**41%**
	Latency Reduction - OTF	**77%**	**68%**	**56%**
	MSF	4.84 ± 0.2		
	OTF	17.6 ± 0.4	11.4 ± 0.4	7.1 ± 0.2
1/4	E-OTF	4.8 ± 0.02	4.3 ± 0.04	3.3 ± 0.1
	Latency Reduction - MSF	**1%**	**11%**	**32%**
	Latency Reduction - OTF	**73%**	**62%**	**54%**
	MSF	5.1 ± 0.3		
	OTF	15.5 ± 1.0	12.6 ± 0.8	5.8 ± 0.1
1/6	E-OTF	6.0 ± 0.1	5.3 ± 0.1	4.1 ± 0.1
	Latency Reduction - MSF	**0%**	**0%**	**20%**
	Latency Reduction - OTF	**61%**	**58%**	**29%**

disruptive events in the network. Fig. 19 and Fig. 20 show the results obtained for the three highest traffic rates (and with $T = 2$ for E-OTF and OTF; the results with different values of T exhibit the same trend and are, thus, omitted). The number of TSCH de-synchronizations is not shown, for the sake of brevity, as these events occur rarely (i.e., 2-3 occurrences per hour, in our simulation experiments). The simulation results in Fig. 19 show that the number of parent changes (due to RPL) is approximately the same for MSF, E-OTF and OTF. Instead, from Fig. 20 it emerges that E-OTF experiences a larger number of schedule mismatches. This is because E-OTF triggers more frequent adjustments in the schedule and, hence, it originates a larger number of 6P transactions. This, in turn, increases the probability of experiencing a schedule mismatch caused by a lost/corrupted 6P message.

In order to provide an insight on the behavior of E-OTF and better understand the reasons of its better performance with respect to MSF and OTF, we compare the queue occupancy, over time, of OTF, MSF and E-OTF (see Fig. 16 and Fig. 18, respectively). We focus on node 7 and consider a single run in which the traffic rate is 2 packets/slotframe. For all the considered SFs, the queue size fluctuates over time, however, its dynamics is quite different in the two cases, as shown in Fig. 18. With E-OTF, after minute 25 a change of parent node occurs, which causes an increase in the number of packets in the queue. This congestion, however, is of very short duration and disappears after approximately one minute. Specifically, as soon as the number of packets in the queue goes beyond the congestion threshold, the node allocates the *bonus* that allows it to progressively reduce the queue level and recover from congestion. Later on, around minute 30, two successfully disruptive events occur, causing the queue to grow up again and, then, to decrease. From the analysis of our simulation trace, we found that this last congestion is caused by a schedule mismatch that forces

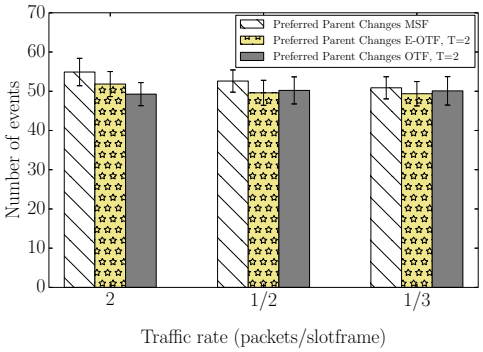


Fig. 19. Preferred Parent Changes.

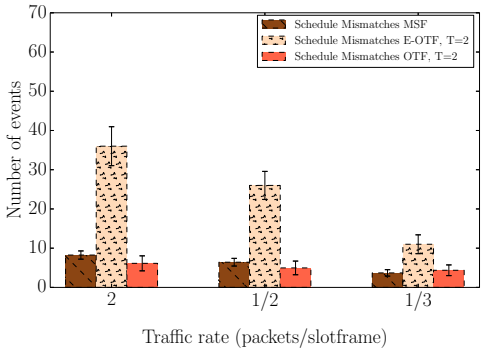


Fig. 20. Schedule Mismatches.

the node to reset its current schedule and start a new negotiation process. In the meanwhile, the node has no allocated cells and, consequently, the queue grows up. With MSF, on the other hand, we can notice that even though the queue at node 7 does not reach the maximum size, after an initial parent change the queue occupancy grows up significantly and remain very high for about 15 minutes. Although the congestion is then recovered, such a high number of enqueued packets results in larger delays and dropped packets. These results show that, while it is not possible to prevent disruptive events, such as parent changes and schedule mismatches, as they are caused by other protocols, E-OTF is able to manage these events more effectively than MSF (and OTF), by reacting promptly and recovering very quickly from congestion.

To analyze the efficiency of the three SFs in allocating the proper number of cells, we considered the average *cell utilization*, shown in Fig. 21 for three traffic rates (2, 1/3, 1/6 packets/slotframe). The results show that E-OTF provides a better utilization in comparison to MSF. This can be explained considering that E-OTF introduces a rapid mechanism to handle congestion, the congestion bonus. The bonus is acquired and released rapidly as the congestion arises and is resolved, thus minimizing the time in which the allocated resources remain unused. On the other hand, MSF adopts a slower approach to release cells when the utilization drops below the lower threshold (25% of current allocation), as only one cell at a time is removed from the schedule. If we compare OTF and E-OTF, we can notice that OTF always results in a better utilization. However, this is because OTF handles congestion poorly, thus resulting in long periods with a high number of enqueued packets that are consequently transmitted in all the available cells. Focusing on E-OTF, we can observe that the difference between the configurations with $T = 2$ and $T = 8$ increases as the traffic rate decreases, i.e., the utilization decreases when a larger T value is adopted. This is because at low traffic rate, a large congestion bonus allows the nodes to eliminate congestion right after the allocation of the bonus cells, which, however, remain unused until the subsequent execution of E-OTF. We conclude our discussion about this set of experiments, by analyzing the *duty cycle* of a node when using the three considered SFs. It measures the percentage of time in which the node remains active. At the same time, if we assume that nodes remain active only during the cells in which they transmit or receive (and sleep for the rest of the time, to save energy), the duty cycle is also an indirect measure of the energy consumed by the node. Fig. 22 shows the duty cycle experienced by a specific node in the network (node 7), for three different traffic rates, i.e., high: 2 packets/slotframe, moderate: 1 packet/slotframe, low: 1 packets/6 slotframe. For the sake of brevity, we only considered two different values of the hysteresis quantum T, namely $T = 2$ and $T = 8$ (the results obtained with $T = 4$ has the same trend). As above, we focus on node 7 since it is one of the most loaded nodes in

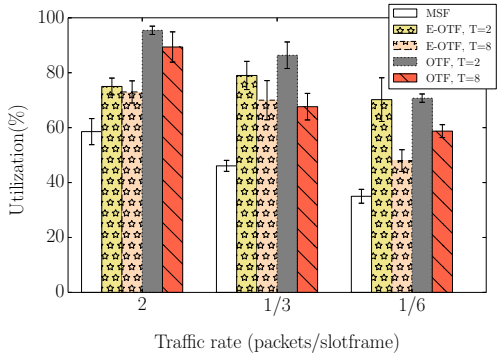


Fig. 21. Cell Utilization.

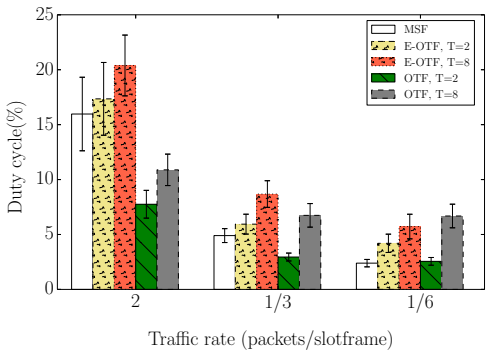


Fig. 22. Duty Cycle of node 7.

the network, due to its location in the grid (see Fig. 9). E-OTF exhibits a duty cycle slightly higher than MSF and higher than OTF. Once again, this is due to the congestion bonus, i.e., the extra number of cells requested by the algorithm to quickly recover from congestion, that increases the overall duty cycle. However, the additional cost to pay with respect to MSF, is limited and acceptable, compared with the significant improvement introduced by E-OTF in terms of lower end-to-end latency and higher reliability. The low duty cycle of OTF, especially with 2 packets/slotframe, is due to the previously highlighted shortcomings of its Allocation Algorithm. Indeed, the duty cycle is low because the number of required cells is underestimated which results in a low end-to-end reliability (Fig. 17).

So far, we have compared the performance of E-OTF, MSF, and OTF for varying traffic rates. In practice, many other parameters may affect the performance of the SF, such as slotframe size, number of nodes in the network, etc. To investigate the effect of such parameters, we performed additional simulation experiments. The obtained results are summarized below. For brevity, we only focus on E-OTF (with $T = 2$) and MSF.

Fig. 23 shows the end-to-end reliability for different slotframe sizes, when the traffic rate is equal to 1 packet every 1.5 seconds (i.e., 1 packet/slotframe when the slotframe size is equal to 101). The performance of the SFs is not significantly affected by the slotframe size, confirming that the considered SFs work well with different configurations. The latency experienced by packets,

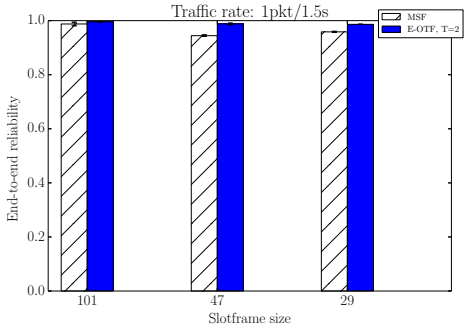


Fig. 23. End-to-end reliability vs slotframe size.

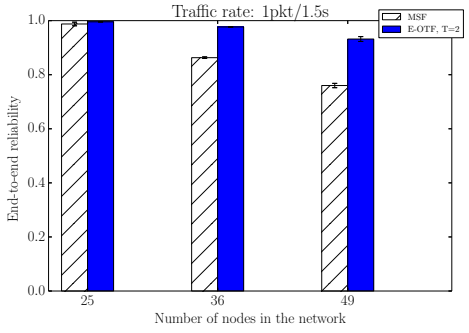


Fig. 24. End-to-end reliability vs topology size.

omitted here for brevity, decreases significantly with the slotframe size. This is mainly because when the slotframe size is lower, the allocated cells are more uniformly distributed over time.

Fig. 24 shows the impact of the number of nodes in the network, (as above, the traffic rate is 1 packet/slotframe, corresponding to 1 packet every 1.5 seconds). Now, the impact on the performance of both E-OTF and MSF is much more apparent since the path from the source node to the destination node is typically longer. Hence, the end-to-end reliability decreases and the latency (omitted here for brevity) increases. However, E-OTF still outperforms MSF in all the considered scenarios and the gap increases with the network size.

9 EXPERIMENTAL RESULTS

Since simulation experiments might not take into account all the factors that can occur in a real environment, we also performed a set of measurements on a real testbed. The purpose of this experimental analysis is twofold: (i) validating the previous simulation results, thus confirming that E-OTF outperforms both MSF and OTF, and (ii) showing that the proposed solution is viable in a real environment. For our measurements, we used the PINT testbed [32] deployed on a two-floor building of our department at the University of Pisa, whose map is shown in Fig. 25. It consists of 23 nodes, where each node is a Zolertia RE-Mote sensor node, equipped with ARM Cortex-M3 system on chip (SoC), and running the Contiki Operating System. Wireless connectivity is provided by the Texas Instruments CC2538 chip, operating at 2.4 GHz, that implements the IEEE 802.15.4 PHY layer. As for the simulation comparison in Section 8, we limited our experimental analysis to E-OTF, MSF and OTF. In our experiments, we used the same Contiki code used in the Cooja simulator and the same parameter values considered in the simulation analysis (see Tab. 4), with a couple of exceptions. Specifically, we reduced the size of the slotframe to 47 slots and limited the buffer available at nodes to 8 packets, in order to meet the memory constraints of the real sensor nodes used in our testbed.

Although we did not carry out our measurements in an industrial environment, nevertheless we ran our experiments in a working place with several sources of interfering signals, including co-located WiFi networks. To increase the accuracy of our results, we performed 10 different replicas for each experiment, each spanning 1 hour. The results presented below are averaged over the 10 replicas (in the figures we also show the confidence intervals). We considered the following traffic rates: 1, 1/2, 1/3, 1/6 packets/slotframe.

Fig 26 compares the end-to-end reliability provided by MSF, OTF and E-OTF. The trend is the same observed in simulation experiments (see Fig. 17), with E-OTF significantly outperforming OTF and MSF, at the higher traffic rates, also in a real environment. Specifically, E-OTF provides a reliability above 90% for all the considered traffic rates, and very close to 100% when the traffic

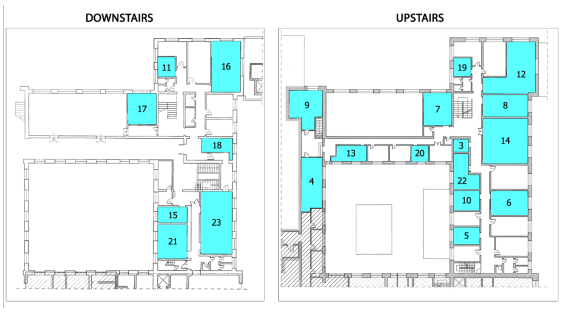


Fig. 25. PINT Testbed Deployments map.

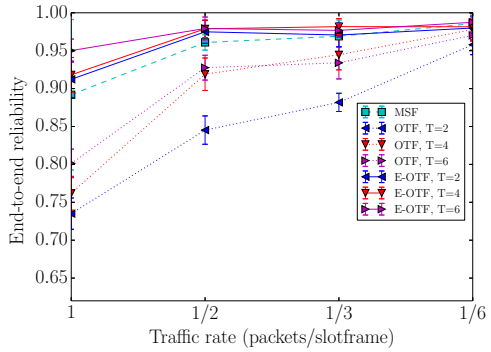


Fig. 26. End-to-end reliability in the PINT testbed.

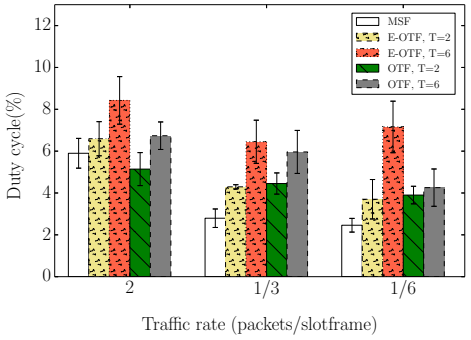


Fig. 27. Duty Cycle of node 3 in the PINT testbed.

Table 7. 95^{th} Percentile end-to-end Latency(s). PINT testbed results.

T (Hysteresis Quantum)		2	4	6
packets/slotframe	MSF	4.7 ± 0.3		
1	OTF	7.5 ± 0.6	6.3 ± 0.2	4.9 ± 0.24
	E-OTF	3.3 ± 0.3	2.9 ± 0.1	3.5 ± 0.5
	Latency Reduction - MSF	**30%**	**38%**	**26%**
	Latency Reduction - OTF	**56%**	**54%**	**29%**
1/2	MSF	5.1 ± 0.2		
	OTF	7.4 ± 0.7	4.1 ± 0.3	3.1 ± 0.2
	E-OTF	3.3 ± 0.2	2.8 ± 0.2	2.5 ± 0.03
	Latency Reduction - MSF	**35%**	**45%**	**51%**
	Latency Reduction - OTF	**55%**	**32%**	**19%**
1/3	MSF	6 ± 0.3		
	OTF	5.9 ± 0.7	4.3 ± 0.2	3.4 ± 0.2
	E-OTF	4 ± 0.1	3.3 ± 0.1	2.6 ± 0.04
	Latency Reduction - MSF	**33%**	**45%**	**43%**
	Latency Reduction - OTF	**32%**	**23%**	**24%**
1/6	MSF	5.7 ± 0.2		
	OTF	4.9 ± 0.1	4.1 ± 0.1	3.5 ± 0.3
	E-OTF	4.4 ± 0.1	3.8 ± 0.07	3.2 ± 0.1
	Latency Reduction - MSF	**23%**	**33%**	**44%**
	Latency Reduction - OTF	**10%**	**7%**	**9%**

rate is lower than 1 packet/slotframe. Tab 7 shows the 95^{th} percentile of the end-to-end latency experienced by packets. As above, the trend in latency reduction is the same observed in simulation experiments, with E-OTF significantly outperforming both OTF (up to 56%) and MSF (up to 51%).

Now, if we consider the gain obtained with E-OTF over MSF and OTF, the absolute values of the real experiments are lower than in simulations. This is due to the different configuration used in the real experiments to meet the memory constraints, i.e., the smaller buffer available on the nodes (16 in simulations and 8 in real experiments) and the shorter slotframe length (101 in simulations and 47 in real experiments).

To conclude the analysis of our experimental results with the PINT testbed, in Fig. 27 we show the duty cycle of node 3. Node 3 has a central position in the network and, hence, it has many neighbors and a large amount of traffic to manage. It can be seen that, when the traffic rate is 1 packet/slotframe, E-OTF shows values that, with $T = 2$, are comparable with MSF. Again OTF has a lower duty cycle for this traffic rate, coherent to the low end-to-end reliability reported in Fig 26. With the other traffic rates, E-OTF with $T = 2$ exhibits a duty cycle which is around 2% higher than MSF, and this gap increases with $T = 6$, nevertheless allowing the end-to-end latency to be the lowest.

10 CONCLUSIONS

In this paper we have performed a comprehensive performance evaluation of the 6TiSCH distributed resource management mode, by considering not only the SF but also its interplay with the RPL routing protocol and the 6P protocol used for resource negotiation. Firstly, we have derived an analytical model of the 6P protocol to calculate the time required to complete a negotiation and its success/failure probability. Then, we have investigated, through simulation, the performance of three distributed SFs, namely MSF, OTF and PID, that take different approaches in managing communication resources. The results obtained have shown that, the behavior of the considered SFs is strongly impacted by frequent changes in the parent node introduced by RPL, as well by failures of 6P transactions. Both events may cause congestion at nodes, resulting in low reliability and high end-to-end latency experienced by packets. In order to overcome these shortcomings, we have proposed an enhanced version of OTF, namely E-OTF. The proposed solution leverages a mechanism that allows nodes to react to congestion rapidly, resulting in a significant improvement in terms of reliability and latency. Finally, we have also performed an experimental analysis using a real testbed to validate our simulation results and assess the suitability of the proposed enhancements in a real environment. The experimental measurements confirm the simulation results and show that E-OTF outperforms both MSF and OTF, in terms of both latency and reliability at the cost of a slightly higher duty cycle. As future work, we plan to design a new Scheduling Function that takes a hybrid approach, by combining static and dynamic resource allocation, in order to ensure continuous data forwarding and further reducing end-to-end latency.

11 ACKNOWLEDGMENTS

This work was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

REFERENCES

[1] 2016. IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)* (April 2016). https://doi.org/10.1109/IEEESTD.2016.7460875

[2] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler. 2015. Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation. *IEEE Internet of Things Journal* (2015). https://doi.org/10.1109/JIOT.2015.2476915

[3] A. Aijaz and U. Raza. 2017. DeAMON: A Decentralized Adaptive Multi-Hop Scheduling Protocol for 6TiSCH Wireless Networks. *IEEE Sensors Journal* (2017). https://doi.org/10.1109/JSEN.2017.2746183

[4] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. Dujovne. 2019. *6TiSCH Minimal Scheduling Function (MSF).* Internet-Draft. https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-msf-05 Work in Progress.

[5] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana. 2016. LLSF: Low Latency Scheduling Function for 6TiSCH Networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS).* https://doi.org/10.1109/DCOSS.2016.10

[6] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. 2005. A High-throughput Path Metric for Multi-hop Wireless Routing. *Wireless Networks* (2005). http://dx.doi.org/10.1007/s11276-005-1766-z

[7] D. De Guglielmo, B. Al Nahas, S. Duquennoy, T. Voigt, and G. Anastasi. 2017. Analysis and Experimental Evaluation of IEEE 802.15.4e TSCH CSMA-CA Algorithm. *IEEE Transactions on Vehicular Technology* (2017). https://doi.org/10.1109/TVT.2016.2553176

[8] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne. 2016. Distributed PID-Based Scheduling for 6TiSCH Networks. *IEEE Communications Letters* (2016). https://doi.org/10.1109/LCOMM.2016.2546880

[9] D. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura. 2018. *6TiSCH Experimental Scheduling Function (SFX).* Internet-Draft. https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-6top-sfx-01

[10] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In *ACM Conference on Embedded Networked Sensor Systems (SenSys '15).* https://doi.org/10.1145/2809695.2809714

[11] S. Duquennoy, A. Elsts, B. Al Nahas, and G. Oikonomou. 2017. TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation. In *International Conference on Distributed Computing in Sensor Systems (DCOSS).* https://doi.org/10.1109/DCOSS.2017.29

[12] A. Elsts, X. Fafoutis, J. Pope, G. Oikonomou, R. Piechocki, and I. Craddock. 2017. Scheduling High-Rate Unpredictable Traffic in IEEE 802.15.4 TSCH Networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. https://doi.org/10.1109/DCOSS.2017.20

[13] A. Elsts, S. Kim, H. Kim, and C. Kim. 2020. An Empirical Survey of Autonomous Scheduling Methods for TSCH. *IEEE Access* (2020). https://doi.org/10.1109/ACCESS.2020.2980119

[14] X. Fafoutis, A. Elsts, G. Oikonomou, R. Piechocki, and I. Craddock. 2018. Adaptive static scheduling in IEEE 802.15.4 TSCH networks. In *IEEE World Forum on Internet of Things (WF-IoT)*. https://doi.org/10.1109/WF-IoT.2018.8355114

[15] D. Fanucchi, B. Staehle, and R. Knorr. 2018. Network Formation for Industrial IoT: Evaluation, Limits and Recommendations. In *IEEE International Conference on Emerging Technologies and Factory Automation (EFTA)*. https://doi.org/10.1109/ETFA.2018.8502509

[16] I. Hosni and F. Théoleyre. 2017. Self-healing distributed scheduling for end-to-end delay optimization in multihop wireless networks with 6TiSCh. *Computer Communications* (2017). https://doi.org/10.1016/j.comcom.2017.05.014

[17] S. Jeong, H. Kim, J. Paek, and S. Bahk. 2020. OST: On-Demand TSCH Scheduling with Traffic-Awareness. In *IEEE International Conference on Computer Communications, INFOCOM 2020*.

[18] S. Jeong, J. Paek, H. Kim, and S. Bahk. 2019. TESLA: Traffic-Aware Elastic Slotframe Adjustment in TSCH Networks. *IEEE Access* (2019). https://doi.org/10.1109/ACCESS.2019.2940457

[19] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara. 2016. A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks. In *IEEE Wireless Communications and Networking Conference*. https://doi.org/10.1109/WCNC.2016.7565002

[20] Y. Jin, U. Raza, A. Aijaz, M. Sooriyabandara, and S. Gormus. 2018. Content Centric Cross-Layer Scheduling for Industrial IoT Applications Using 6TiSCH. *IEEE Access* (2018). https://doi.org/10.1109/ACCESS.2017.2762079

[21] A. Karaagac, I. Moerman, and J. Hoebeke. 2018. Hybrid Schedule Management in 6TiSCH Networks: The Coexistence of Determinism and Flexibility. *IEEE Access* (2018). https://doi.org/10.1109/ACCESS.2018.2849090

[22] S. Kim, H.-S. Kim, and C. Kim. 2019. ALICE: Autonomous Link-based Cell Scheduling for TSCH. In *International Conference on Information Processing in Sensor Networks (IPSN '19)*. http://doi.acm.org/10.1145/3302506.3310394

[23] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne. 2013. Label switching over IEEE802.15.4e networks. *Transactions on Emerging Telecommunications Technologies* (2013). https://doi.org/10.1002/ett.2650

[24] E. Municio and S. Latré. 2016. Decentralized Broadcast-based Scheduling for Dense Multi-hop TSCH Networks. In *Workshop on Mobility in the Evolving Internet Architecture (MobiArch '16)*. http://doi.acm.org/10.1145/2980137.2980143

[25] E. Municio, K. Spaey, and S. Latré. 2018. A distributed density optimized scheduling function for IEEE 802.15.4e TSCH networks. *Transactions on Emerging Telecommunications Technologies* (2018). https://doi.org/10.1002/ett.3420

[26] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. 2006. Cross-Level Sensor Network Simulation with COOJA. In *IEEE Conference on Local Computer Networks*. https://doi.org/10.1109/LCN.2006.322172

[27] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel. 2016. On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks. *IEEE Sensors Journal* (2016). https://doi.org/10.1109/JSEN.2015.2480886

[28] F. Righetti, C. Vallati, G. Anastasi, and S. K. Das. 2017. Performance Evaluation the 6top Protocol and Analysis of its Interplay with Routing. In *IEEE International Conference on Smart Computing (SMARTCOMP)*. https://doi.org/10.1109/SMARTCOMP.2017.7947029

[29] F. Righetti, C. Vallati, G., and S. K. Das. 2018. Analysis and Improvement of the On-The-Fly Bandwidth Reservation Algorithm for 6TiSCH. In *IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. https://doi.org/10.1109/WoWMoM.2018.8449793

[30] P. Thubert. 2019. *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4*. Internet-Draft. https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-architecture-24 Work in Progress.

[31] P. Thubert, M. R. Palattella, and T. Engel. 2015. 6TiSCH centralized scheduling: When SDN meet IoT. In *IEEE Conference on Standards for Communications and Networking (CSCN)*. https://doi.org/10.1109/CSCN.2015.7390418

[32] C. Vallati, E. Ancillotti, R. Bruno, E. Mingozzi, and G. Anastasi. 2016. Interplay of Link Quality Estimation and RPL Performance: An Experimental Study. In *ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, PE-WASUN*. http://doi.acm.org/10.1145/2989293.2989299

[33] C. Vallati, S. Brienza, G. Anastasi, and S. K. Das. 2019. Improving network formation in 6TiSCH networks. *IEEE Transactions on Mobile Computing* (2019). https://doi.org/10.1109/TMC.2018.2828835

[34] X. Vilajosana, K. Pister, and T. Watteyne. 2017. Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration. RFC 8180. https://doi.org/10.17487/RFC8180

[35] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, and P. Thubert. 2020. IETF 6TiSCH: A Tutorial. *IEEE Communications Surveys Tutorials* (2020). https://doi.org/10.1109/COMST.2019.2939407

[36] Q. Wang, X. Vilajosana, and T. Watteyne. 2018. 6TiSCH Operation Sublayer (6top) Protocol (6P). RFC 8480. https://doi.org/10.17487/RFC8480