IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2021, accepted September 2, 2021, date of publication September 13, 2021, date of current version September 21, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3112266*

# Performance Evaluation of Adaptive Autonomous Scheduling Functions for 6TiSCH Networks

**FRANCESCA RIGHETTI, (Member, IEEE), CARLO VALLATI, (Member, IEEE), ARIANNA GAVIOLI, AND GIUSEPPE ANASTASI, (Member, IEEE)**
Department of Information Engineering, University of Pisa, 56126 Pisa, Italy

Corresponding author: Francesca Righetti (francesca.righetti@ing.unipi.it)

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

**ABSTRACT** The Internet Engineering Task Force (IETF) has recently defined the 6TiSCH architecture to enable the *Industrial Internet of Things* (IIoT), i.e., the adoption of the IoT paradigm for industrial applications with stringent requirements, in terms of reliability and timeliness. In 6TiSCH networks, the scheduling of communication resources is of paramount importance to meet the application requirements, and many different *Scheduling Functions* have been proposed to cope with the needs of various applications. Recently, autonomous scheduling has emerged as an efficient and robust approach, as it allows nodes to allocate communication resources *autonomously*, i.e., without any negotiation with their neighbors, thus avoiding the related overhead. Typically, this is obtained through *static* resource-allocation algorithms that are not able to adapt to variations in traffic conditions. In this paper, we consider *adaptive autonomous scheduling*, and compare the performance of three different algorithms in various IIoT scenarios. We investigate their ability to adapt to traffic changes, and evaluate them in terms of performance, resource consumption, and complexity. Based on the results obtained, we also provide a set of guidelines to select the most appropriate Scheduling Function, and its configuration parameters, depending on the specific use case.

**INDEX TERMS** 6TiSCH architecture, autonomous scheduling, adaptability, Industrial Internet of Things, simulation.

## I. INTRODUCTION

The *Industrial Internet of Things (IIoT)* is an evolution of the traditional *Internet of Things* to include industrial applications with stringent requirements, in terms of reliability and timeliness. It is expected to introduce a radical change in the way industrial systems are designed and managed, and to affect many application domains, including manufacturing and production systems, logistics, transportation, energy, and many others [1].

To support this trend, the Internet Engineering Task Force (IETF) has defined the 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) architecture that allows to integrate IoT devices into existing IPv6 networks, while ensuring the stringent requirements of industrial applications [2], [3]. To achieve this goal, 6TiSCH is built on top of the *Time Slotted Channel Hopping* (TSCH) mode of operation of the IEEE 802.15.4 standard [4], which provides time-bounded, guaranteed-bandwidth, and energy-efficient communication through *time-slotted access*, high network capacity through *multi-channel communication*, and robustness against interference and fading through *frequency hopping*.

In 6TiSCH, an appropriate communication scheduling is needed to meet the stringent requirements of industrial applications. Hence, the core of the 6TiSCH architecture is the *Scheduling Function* (SF) used to schedule communication resources (i.e., TSCH cells). Indeed, the 6TiSCH specifications include a *Minimal Scheduling Function* (MSF) [5], that can be assumed as the default SF for 6TiSCH networks. However, other SFs can be used to cope with the requirements of specific use cases. Hence, a large number of SFs have been proposed in the literature ([6]–[15]).

The proposed SFs can be broadly classified according to the approach they take to allocate TSCH cells to nodes, namely, *centralized*, *distributed*, and *autonomous*. In centralized scheduling, a specific node collects information about the network topology and traffic patterns, derives the communication schedule, and communicates it to all the other

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

nodes. Instead, in distributed scheduling, cells are negotiated by neighbor nodes, based on their traffic needs, through a specific negotiation protocol. Finally, in autonomous scheduling cells are allocated by nodes *autonomously*, i.e., without any negotiation.

In this paper, we focus on autonomous scheduling [16], which has emerged recently as an efficient and robust solution for the IIoT. Specifically, the autonomous allocation of cells by nodes does not require any exchange of control packets, thus avoiding negotiation overhead, inconsistencies between neighbor nodes, as well as vulnerability to possible attacks from malicious nodes during the negotiation process [17]. For instance, the Contiki operating system[1] implements an autonomous algorithm as its default SF, namely Orchestra [11].

Autonomous scheduling algorithms traditionally leverage a *static allocation* scheme [11], [12], and, hence, they are unable to adapt to traffic changes. Recently, a number of *adaptive* autonomous SFs have been proposed [14], [15], [18], that can adapt to traffic changes, without exchanging control packets and, hence, with minimal, or null, negotiation overhead. MSF can also be regarded as an adaptive autonomous SF, as it uses a basic amount of autonomous cells, for control traffic only, and allocates additional cells for data packets dynamically. The latter ones are negotiated using a neighbor-to neighbor approach.

In the context of autonomous adaptive SFs, it is not easy to select the best available option, as different SFs take different approaches to adaptation and are characterized by different performance and complexity. A comparative evaluation of different adaptive SFs in different and representative scenarios is still missing, as most of them have been proposed only recently.

In this paper, we consider three adaptive autonomous SFs, namely OST [15], ALICE [12] with FP extension (ALICE-FP), and MSF [5], and perform a comprehensive simulation study to investigate their suitability to real-world use cases. To this end, we analyze four representative IoT scenarios, characterized by different communication paradigms (*many-to-one*, *one-to-many*, *one-to-one*) and traffic patterns (*periodic* or *bursty*), and evaluate the considered SFs, in terms of *reliability*, *timeliness*, *resource consumption*, and *complexity*. The paper is a continuation of our previous analysis published in [18], where we investigated the pros and cons of neighbor-to-neighbor negotiation vs. autonomous allocation. We observed that a static resource allocation may be a serious limitation for autonomous scheduling. In this work, we focus on *adaptive autonomous scheduling* in order to carry out a comprehensive evaluation of the most recent adaptive autonomous SFs proposed in literature. Our results have shown that none of the considered SFs outperforms the other ones in all the considered scenarios. Instead, different SFs exhibit pros and cons under different conditions. Basically, OST optimizes the resource consumption (i.e., energy and bandwidth consumption), at the cost of increased end-to-end delay, while ALICE-FP takes the opposite approach and favors timeliness, at the cost of a higher energy/bandwidth consumption. Also, the different SFs have different complexity and react differently to changes in the operating conditions. Hence, deciding the most suitable SF, for the specific use case, may not be an easy task. Therefore, in the last part of the paper, we provide a set of guidelines that can help the designer of IoT systems in the selection of the SF, depending on the application requirements and operating conditions.

In conclusion, the main contributions provided by this paper can be summarized as follows:

- A comparison of three adaptive (autonomous) SFs for 6TiSCH networks in four representative IoT scenarios, characterized by different communication paradigms (*many-to-one*, *one-to-many*, *one-to-one*) and traffic patterns (*periodic* or *bursty*);
- A comprehensive evaluation of the considered SFs, in terms of *performance* (reliability and timeliness), *resource consumption* (duty cycle), and *complexity* (control bits);
- A set of guidelines to select the most appropriate SF, depending on the specific use case and operating conditions.

The reminder of this paper is organized as follows. In Section II, we describe the 6TiSCH architecture. In Section III, we provide a general overview of SFs for 6TiSCH and the related work, with special focus on the three SFs considered in our analysis. In Section IV, we present our simulation methodology. In Section V, we compare the three SFs in terms of reliability, timeliness and resource consumption, while in Section VI we analyze their complexity. In Section VII, we summarize the lessons learned from our study and provide a set of guidelines for selecting the "best" SF, depending on the specific scenario. Finally, in Section VIII we conclude the paper.

## II. 6TiSCH ARCHITECTURE

The 6TiSCH architecture [2] aims at integrating wireless networks based on the 802.15.4 TSCH standard [4] into existing IPv6 infrastructures. The reference architecture and the complete protocol stack are shown in Figure 1.

At the MAC layer, the TSCH protocol allows wireless communication with bounded delay, high reliability, and low energy consumption. To this end, TSCH relies on *time-slotted channel access*, *multi-channel communication*, and *frequency hopping*. Time is divided into time intervals of fixed duration (timeslots), each of which allows the transmission of a packet and the corresponding acknowledgment. A number of consecutive timeslots form a *slotframe*, which repeats periodically over time. To increase the network capacity, different nodes are allowed to transmit simultaneously on the same timeslot, using a different channel (multi-channel communication). Specifically, 16 different channels are available, identified by a *channel offset* (an integer value in the range 0-15) and, hence, each cell in this two-dimensional slotframe is

[1] https://github.com/contiki-ng/contiki-ng

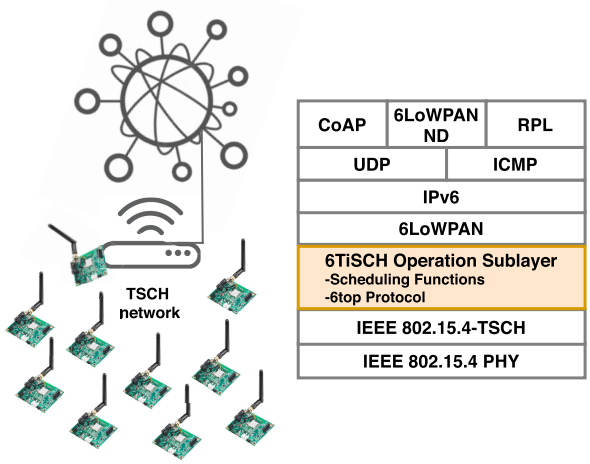



**FIGURE 1.** 6TiSCH reference architecture and protocol stack.

identified through a couple of information, namely *timeslot*, and *channel offset*. Finally, to mitigate the negative effects of multipath fading and interferences, TSCH leverages frequency hopping. A predefined frequency-hopping sequence is shared among all the nodes in the network, so that they can select a different operating frequency at each timeslot.

The TSCH protocol provides mechanisms to allocate and deallocate cells to network nodes, according to a communication schedule. Among these mechanisms, the TSCH MAC frame includes the *Frame Pending* (FP) flag, a bit within the Frame Control Field, that can be used by the sending device to signal that it has more data for the recipient. When set to one, the FP bit indicates that the recipient should remain active in the next timeslot and on the same channel, unless the next cell is already allocated for other purposes.

While TSCH provides mechanisms to allocate and deallocate cells, it does not specify how cells are allocated to nodes for communication. To this purpose, the 6TiSCH architecture includes the 6TiSCH Operation (6top) sublayer [19] that provides the abstraction of IP link over TSCH, by managing the allocation of cells to nodes in such a way to meet the application requirements.

Above the 6top layer, the 6LoWPAN adaptation protocol is responsible for encapsulating IPv6 datagrams into TSCH frames, while the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) ensures multi-hop delivery of IPv6 datagrams. RPL [20] organizes the network nodes in a *Destination Oriented Directed Acyclic Graph (DODAG)*, where every node selects a neighbor, called *preferred parent*, as the candidate neighbor for upstream data delivery. The DODAG is rooted at a single node, the root node, that is typically the collector of the network to which upstream data is directed. Although RPL is optimized for upstream data delivery, downstream data delivery, from the root node to non-root nodes, is also supported. Finally, the end-to-end delivery of data packets originated by the application is managed by the UDP protocol.

The 6top sublayer [19] is a crucial part of the 6TiSCH architecture, as it determines the schedule used by nodes for communication. It consists of two main components, namely a *Scheduling Function (SF)* to calculate the number of cells to allocate, depending on the current conditions, and the *6top (6P) protocol* to negotiate the required cells, when a neighbor-to-neighbor negotiation is used. The main SFs for 6TiSCH will be discussed in the next Section. We provide below a brief description of the 6P negotiation protocol, that is used by MSF.

6P [19] defines the operations and messages to implement a complete negotiation between two nodes (6P transaction). A 6P transaction consists of a *Request* followed by a *Response*. The Request message includes a code to specify the requested action, namely, ADD (to request a new allocation), DELETE (to cancel an existing allocation), or CLEAR (to reset the current negotiation). Similarly, the Response message includes a SUCCESS or ERROR code, to notify a successful or failed transaction, respectively.

Typically, a requesting node A sends an ADD request to the corresponding node B, specifying the number of cells to allocate and a list of free cells. Then, node B replies with a SUCCESS response containing the list of allocated cells, if available. Otherwise, node B replies with an ERROR code. If a Request/Response message gets corrupted, it is retransmitted after a predefined timeout (6P Timeout). Finally, when a mismatch is detected by node B (e.g., due to a lost Response message), it replies with an ERROR message that forces node A to send a CLEAR message and resets the schedule (both nodes cancel all the allocated cells). Then, node A starts a new 6P transaction.

## III. SCHEDULING FUNCTIONS AND RELATED WORK

In this Section, we provide a classification of the main approaches to scheduling in 6TiSCH networks, with special emphasis on autonomous scheduling. This allows us to discuss also the related work. Then, we will focus on the three SFs considered in our study.

### A. CLASSIFICATION

A significant number of SFs for 6TiSCH networks have been proposed to cope with the requirements of different use cases. They can be broadly classified according to the approach they take to allocate cells to nodes, namely, *centralized*, *distributed*, and *autonomous* scheduling. In addition, there are also *hybrid* solutions that combine some of the previous approaches.

*Centralized scheduling* leverages a central entity, referred to as *Point Coordination Element* (PCE), that collects information about the network topology and traffic patterns and, based on those, derives the (optimal) communication schedule [6]–[8], [21], [22]. Some of these centralized SFs assume an interference-free communication channel, or at least, a limited level of interference. This assumption is not realistic as IIoT networks often coexist with other networks operating on the same frequency band, such as WiFi and

Bluetooth networks, which results in communication unreliability and topology instability. In [7] and [8], the authors explicitly take interference into account and consider retransmission cells in their schedule. Specifically, [7] proposes a link-based retransmission scheme, while [8] leverages a flow-based retransmission policy, which proves to be more efficient. In general, centralized scheduling provides optimal schedule, but requires a high communication overhead, especially in large networks. Also, the optimal schedule must be re-computed whenever the operating conditions change. Hence, centralized scheduling is unsuitable to large and/or dynamic networks, i.e., networks where the operating conditions change often, due to traffic variations, link unreliability, changes in the network topology (e.g., due to the RPL protocol).

In *distributed scheduling* [10], nodes negotiate the allocation of TSCH cells with their neighbors, using the 6P protocol, and adjust the number of allocated cells, depending on traffic and network conditions. Typically, the communication schedule is not optimal, however this approach is very suitable to large and/or dynamic networks. Among distributed SFs, OTF [9] was originally considered by the 6TiSCH WG for standardization as the reference SF. However, OTF has a number of limitations in estimating the number of cells required by the application, as well as in managing congestion (e.g., originated by changes in the RPL preferred parent). These limitations are overcome by E-OTF [10], an enhanced version of OTF.

### B. AUTONOMOUS SCHEDULING

The main drawback of distributed scheduling is the associated negotiation overhead, due to 6P transactions. To avoid this overhead, *autonomous scheduling* [16] was introduced, where nodes allocate cells autonomously (i.e., without negotiation), using a hash function applied to nodes' addresses.

Orchestra [11] was the first proposed autonomous SF for 6TiSCH, and is currently the default SF used in the Contiki operating system. It leverages a node-based approach that can be declined into two different allocation modes, namely, *receiver-based* and *sender-based*. In the receiver-based mode, each node allocates one cell per slotframe to *receive* packets from all its neighbors, while in the sender-based mode each node has only one cell per slotframe to *send* packets to all its neighbors.

ALICE (*Autonomous Link-based Cell Scheduling*) [12] replaces the node-based allocation approach used in Orchestra with a link-based approach, where each node allocates a separate cell for each of its unidirectional links. Hence, each node has many transmission and reception cells as the number of its neighbors. This allows nodes to allocate a number of cells corresponding to the number of links, i.e., to the amount of traffic to manage. Hence, ALICE significantly outperforms both modes of Orchestra [16].

Autonomous scheduling is essentially static, i.e., for each node, the number of available cells per time unit is constant. Hence, autonomous SFs, like Orchestra and ALICE, may be unable to manage efficiently unpredictable changes in traffic conditions [18]. In addition the allocation is, typically, not optimal [15]. To overcome these limitations, adaptive autonomous algorithms have been proposed, where the number of allocated cells is adjusted over time, depending on traffic conditions. Adaptability requires to exchange information among neighbor nodes to adjust the current allocation and, hence, it re-introduces a control overhead. This overhead is the price to pay to achieve adaptability. However, it should be kept as low as possible, in order to not destroy the basic feature of autonomous scheduling.

Information exchange among neighbors to support adaptation can be implemented in different ways. The simplest way is using a *hybrid* approach that combines autonomous scheduling with neighbor-to-neighbor negotiation, based on the 6P protocol. While autonomous scheduling provides a basic number of cells to the node, additional cells can be added and removed dynamically, through 6P negotiation with neighbors, depending on time-varying traffic conditions. This hybrid approach is used by the *Minimal Scheduling Function (MSF)* [5], the reference SF for 6TiSCH networks. With MSF, nodes use both *autonomous* and *negotiated* cells. Autonomous cells provide a minimal amount of bandwidth only for control messages, while negotiated cells are used for managing data traffic.

The main drawback of the hybrid approach is the 6P protocol used for cell negotiation. Several previous studies [10], [23] have pointed out that 6P transactions take a considerable time to complete (in the order of seconds) and may also fail, leading the corresponding nodes to an inconsistent state. In addition, 6P transactions are vulnerable to security attacks that can result in selective jamming of the communication towards a victim node, and/or useless transmissions (and, hence, energy wastage) of the victim node [17]. An alternative approach for exchanging control information among neighbors consists in piggybacking such information in data packets. This reduces the overhead and avoids the drawbacks of using an independent negotiation protocol, like 6P.

In [18], the authors proposed ALICE-FP, an adaptive version of ALICE that leverages the *Frame Pending* (FP) option made available by the underlying TSCH protocol [4]. This allows the sending node to signal its corresponding node that it has more data to send by setting the FP bit in the TSCH frame. Hence, a node can get extra cells, in addition to (*autonomously*) scheduled cells, thus facing temporary traffic peaks. Clearly, the adaptability provided by ALICE-FP is limited; however, it is obtained with *no overhead*, as the FP bit is already present in the underlying TSCH frame.

TESLA [14] and OST [15] also exploit piggybacked information to support adaptability. However, they implement more complex adaptation schemes that require more control information than a single bit. As a consequence, the adaptation-related overhead is not null, as in ALICE-FP; however, it is still limited due to piggybacking.

TESLA (*Traffic-Aware Elastic Slotframe Adjustment*) [14] builds on top of the Orchestra receiver-based version, where

each node allocates a single (reception) slot per slotframe for receiving data packets from all its neighbors, which results in static scheduling with globally identical slotframe. Since network nodes have different and time-varying traffic requirements, TESLA allows each node to self-adjust its slotframe size, depending on its incoming traffic. To this end, each node receives information about its incoming traffic piggybacked in the underlying MAC frame and, hence, without any additional control overhead. Based on this information, the node periodically estimates the contention level of its neighbors, and adjusts its slotframe size accordingly. Specifically, the slotframe size is decreased when the contention level is high (to reduce the collision probability and improve reliability), and increased when it is low (to avoid idle listening and save energy). Upon a slotframe size change, the node communicates the new size to all its one-hop neighbors (i.e., preferred parent and children). This is the price to pay for dynamic adaptation.

OST (*On-demand Scheduling with Traffic awareness*) [15] is more similar to ALICE and implements an adaptive allocation scheme with traffic awareness. It aims at minimizing energy consumption while guaranteeing reliable packet delivery. Specifically, to meet the different traffic requirements efficiently, OST leverages two different allocation mechanisms, namely *Periodic Provisioning* and *On-demand Provisioning*. Periodic Provisioning allocates a periodic cell for each unidirectional link, as in ALICE, but adjusts the period dynamically, according to the average traffic load on the link. Instead, On-demand Provisioning is used to allocate temporarily additional consecutive cells to cope with unexpected traffic bursts. A negotiation procedure between neighbors is necessary to agree on the parameters of the Periodic Provision mechanism (namely, period and time offset). Control information is exchanged through regular data or ACK packets, without any additional control packet. However, the negotiation procedure introduces a certain overhead. In addition, it may fail, due to lack of available resources at the receiver or sender side (see Section III-C for details).

A performance evaluation comparing OST with Orchestra and ALICE was performed in [15], using a 72-node testbed. The results showed that OST outperforms both Orchestra and ALICE, in terms of reliability and energy efficiency. The reason is that Orchestra and ALICE implement static autonomous scheduling, and are thus unable to adapt to changes in traffic conditions, while OST is adaptive. A comprehensive comparison of adaptive autonomous SFs is still missing.

In this paper, we focus on *adaptive autonomous* scheduling and compare three SFs that leverage different approaches to adapt the autonomous scheduling to traffic changes. Specifically, we consider OST as the representative of fully adaptive autonomous SF, ALICE-FP as an example of autonomous SF with limited adaptability but zero overhead, and MSF as an example of hybrid SF combining autonomous scheduling with neighbor-to-neighbor negotiation (MSF is also the default SF for 6TiSCH networks). In the following, we present a brief presentation of the three considered SFs.

### C. ON-DEMAND SCHEDULING WITH TRAFFIC AWARENESS (OST)

OST [15] is a traffic-aware adaptive scheduling technique that manages communication cells separately for each directional link, adjusting the number of cells to the traffic requirements of the link. The goal is to allocate the *right amount of bandwidth*, in order to avoid both under-provisioning, which would result in packet loss, and over-provisioning, which would result in energy wastage.

The adaptive cell allocation process takes a two-step approach, based on *Periodic Provisioning* and *On-demand Provisioning*, respectively. The former mechanism allocates a periodic cell for each unidirectional link, and adjusts the period dynamically, based on the average traffic load experienced by the link. Instead, the On-demand Provisioning mechanism is used to allocate temporarily additional cells to cope with unexpected traffic bursts. Before describing the two mechanisms in detail, it is worthwhile to introduce the different slotframes used in OST.

#### 1) SLOTFRAME ARCHITECTURE

To support robust TSCH and RPL operations under dynamic conditions, OST maintains two control slotframes, namely EB slotframe and RPL slotframe (like in Orchestra and ALICE), both with constant periodicity and offset. Instead, adaptive scheduling is handled through the following three slotframes:

- **Periodic-provisioning Tx Slotframe (PTS)**: is used for unicast transmission to a neighbor. Its periodicity and channel offset are variable, and set by the Periodic Provisioning mechanism. Each node maintains a PTS for each of its neighbors.
- **Periodic-provisioning Rx Slotframe (PRS)**: is analogous to PTS, but it is used for reception.
- **Autonomous Unicast Slotframe (AUS)**: is used for both control and data transmissions; it has constant periodicity and a dedicated channel offset. Each node has one AUS.

Specifically, AUS is used for unicast TSCH/RPL control packets to alleviate the contention on the shared EB and RPL slotframes that handle the broadcast control transmissions from all nodes. In addition, AUS can temporarily handle data traffic when a pair of PTS/PRS is not yet installed for a directional link. For instance, when a node selects a new preferred parent, it uses AUS to send data packets to the parent until a PTS/PRS pair is installed on that link.

To better understand how PTS and PRS work, let us consider a directional link from node A to node B. A maintains a dedicated PTS for node B with a single transmission cell, while B must have a corresponding PRS for node A with a reception cell matching the transmission cell in A's PTS. Node A can adjust dynamically its PTS, based on the traffic demand towards node B, and node B must adjust its PRS

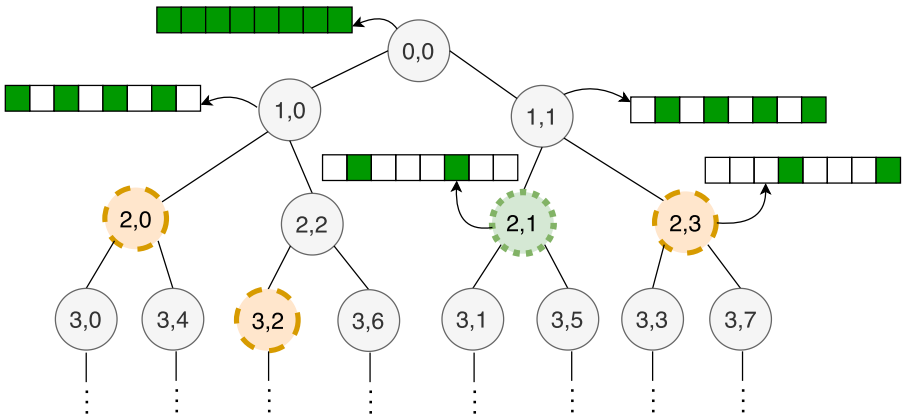



**FIGURE 2.** An example of binary resource tree.

size accordingly. Whenever a parent change occurs, two new PTS/PRS pairs must be installed for the two unidirectional links between the node and its new parent, while the previous PTS/PRS pairs with the old parent are removed.

### 2) PERIODIC PROVISIONING

The Periodic Provisioning mechanism activates periodically, with a period $T$, to adapt the PTS/PRS size, based on the average traffic load measured, during $T$, on the link. The PTS/PRS adaptation procedure occurs at the sender side, since the transmitting node can easily compute the average traffic of the link, and derive the suitable PTS/PRS size. Let us consider a directional link from node A to node B. When the Periodic Provisioning starts, node A computes: (i) the traffic load on the link during the last period $T$, $L(A, B)$, as the number of data packets enqueued for transmission to B during $T$; and (ii) the *Inter-Packet Slots*, *IPS(AB)*, as the total number of cells in $T$ divided by $L(A, B)$. Then, A select the next PTS size as $2^{N(A,B)}$, such that $2^{N(A,B)} \leq IPS(AB) < 2^{N(A,B)+1}$. This means that the sender is given a chance to transmit either with the same frequency, or even with a larger frequency, than the estimated average transmission frequency. Finally, node A piggybacks $N(A, B)$ on the next packets directed to B, so that node B can update its PRS accordingly.

Setting the slotframe size to a power of two allows to exploit a binary resource tree to easily select a cell in the schedule, among the available ones. Each node maintains a binary resource tree and each directional link can use one of the resources in it. Figure 2 shows an example of such a binary resource tree, where nodes with dashed borders (e.g., node (2,3)) designate resources assigned to links. A resource $(n, t)$ in the tree represents a periodic cell with slotframe size $2^n$ and time offset $t$. Each resource $(n, t)$ can be divided equally into two resources, namely $(n + 1, t)$ and $(n + 1, t + 2^n)$. Due to the parent-child relationship, a resource $(n, t)$ always collides with its predecessors and partially collides with the resources in its sub-tree. This characteristic must be considered when selecting a resource for a link in the binary tree, in order to avoid collisions.

Specifically, considering again the generic directional link [A→B], whenever node B detects a new size $N(A, B)$ in a packet from node A, it selects a collision-free resource $(n, t)$ in its binary tree, such that $n = N(A, B)$, and updates its PRS with size $2^{N(A,B)}$ and time offset $t$. The selected time offset is then piggybacked on an ACK for the data packet received from A. It is noteworthy to highlight that, at this point, the resource $(n, t)$ is guaranteed to be collision-free only on the receiving side and that PRS is updated regardless of whether the sender A has an available matching resource or not. When node A receives the time offset $t$ proposed by B, it analyzes its binary resource tree to check whether the resource $(N(A, B), t)$ is available or not. If available, node A simply updates its PTS with size $2^{N(A,B)}$ and with time offset $t$.

Since the selection of the PTS/PRS size and time offset occurs at the sender and receiver side, respectively, two possible allocation failures may occur: (**i**) the receiver B does not have a free resource with the proposed $n = N(A, B)$; or (**ii**) the resource $(N(A, B), t)$, proposed by node B, is not available at node A. In both cases, a negotiation procedure is started.

If there is no available resource with $n = N(A, B)$ in the binary resource tree of the receiver, the latter simply maintains the current PRS configuration and piggybacks an allocation failure code to A. Note that even though node A proposed a new PTS size, the PTS size has not been updated yet. Hence, the PTS and PTR sizes still match. When node A is notified that the node B does not have an available resource with the proposed size, it increments $N(A, B)$ and repeats the whole process again, until A and B find a matching resource.

When the resource proposed by the receiver B is not available at node A, the situation is a little bit more complex. Specifically, node A removes the PTS previously used for transmissions towards B (as node B has already updated its PRS) and sends a failure code to B to notify that the proposed resource is not available, asking for another resource. The negotiation procedure is repeated until a matching time offset is found.

It is important to point out that no valid PTS/PRS pair exists from when node A removes the current PTS to when nodes A and B agree upon a valid time offset. Hence, during this time interval all packets destined to B must be transmitted through the AUS slotframe.

#### 3) ON-DEMAND PROVISIONING

The Periodic Provisioning mechanism aims at allocating the appropriate number of cells to meet the average traffic load. Even if it can adapt to traffic changes, however it reacts slowly, and is thus unable to manage unexpected traffic bursts. Hence, OST also includes the On-demand Provisioning mechanism that allows a node to allocate temporarily additional cells to accommodate extra packets queued in the local buffer.

With reference to the directional link [A→B], whenever A is about to send a data packet towards B, it first checks whether there are other packets queued for B. If this is the case, A computes its *Subsequent Timeslot Schedule (STS)* and piggybacks it on the packet destined to B (that is transmitted with the FP bit set). The STS is an array containing a certain number of bits indicating whether the next cells are allocated or free. To better clarify this point, let us assume that the current *Absolute Slot Number* is $ASN = t^*$. Then, in the STS, the $k^{th}$ bit will be set to 1 if the cell with $ASN = t^* + k$ is already reserved for a scheduled communication, and will be set to 0 otherwise.

When node B receives the packet, it computes its own STS, in turn. Then, node B compares the two arrays and selects the earliest common 0. If this occurs to the $m^{th}$ bit, it means that both A and B have a free cell at $ASN = t^* + m$ and, therefore, that cell can be used to accommodate an extra transmission for the link [A→B]. Then, B piggybacks the position $m$ in the ACK of the received packet, and schedules a temporary receiving cell in $ASN = t^* + m$. Similarly, upon receiving the ACK, node A schedules a transmission in the same cell.

The On-demand Provisioning mechanism is activated repeatedly, whenever there are one or more queued packets. As for Periodic Provisioning, all the information exchanged between nodes is piggybacked on data or ACK packets and, therefore, the mechanism does not require additional control packets. The overhead is limited to the size of STS (e.g., 2 bytes).

### D. ALICE-FP

ALICE [12] was originally proposed to enhance Orchestra [11], especially in large and/or dense networks. Like Orchestra (and OST), ALICE decouples data transmissions from control packet transmissions, using separate channels with dedicated channel offsets. Specifically, it uses an *EB slotframe* for TSCH control packets, an *RPL slotframe* for RPL control packets, and a *unicast slotframe* for data packets. However, ALICE replaces the *node-based* allocation scheme used in Orchestra with a *link-based* scheme, where each node allocates a cell in the unicast slotframe for each unidirectional link. In addition, ALICE re-calculates the cell schedule at every slotframe to minimize the impact of possible conflicts (see below). An example of ALICE unicast slotframe is shown in Figure 3.

Let us consider a directional link from node A to node B, namely [A→B]. Both node A and node B, autonomously, will allocate a cell $(t_{offset}, c_{offset})$ in the unicast slotframe, whose time offset and channel offset are derived as follow:

$$t_{offset}(A,B) = h(\alpha * ID(A) + ID(B))\%S_{sf} \quad (1)$$

$$c_{offset}(A,B) = h(\alpha * ID(A) + ID(B))\%(L_{CH} - 1) + 1 \quad (2)$$

where h is a hash function, $\alpha$ is used to differentiate the traffic direction, $S_{sf}$ is the unicast slotframe length, and $L_{CH}$ is the number of channel offsets used within the unicast slotframe. It may happen that the hash function returns the same value for different links, resulting in a conflict. Since this is unavoidable, ALICE re-computes the cell scheduling at each slotframe to minimize the negative impact of conflicts. Cell rescheduling can be easily obtained by introducing the Absolute Slotframe Number (ASFN) in the computation of the time offset and channel offset, as follows:

$$ASFN = \gamma = \lfloor ASN/S_{sf} \rfloor \quad (3)$$

$$t_{offset}(A,B) = h(\alpha * ID(A) + ID(B) + \gamma)\%S_{sf} \quad (4)$$

$$c_{offset}(A,B) = h(\alpha * ID(A) + ID(B) + \gamma)\%(L_{CH} - 1) + 1 \quad (5)$$

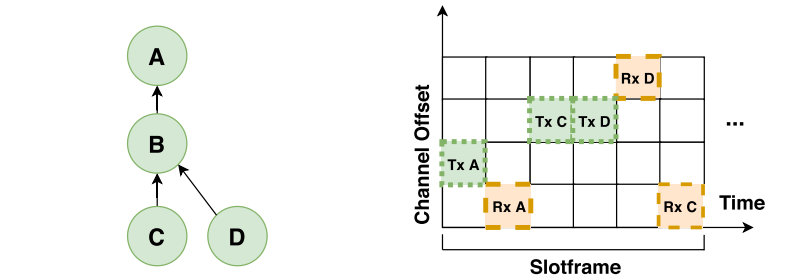
The allocation scheme used in ALICE is static and, hence, it is not able to manage efficiently temporary bursts of packets. Specifically, for each unidirectional link, there is one reserved cell per slotframe. If the number of packets to be transmitted on the link exceeds temporarily this fixed amount of bandwidth, packets must be buffered in a local queue. This results in increased latency and, in extreme cases, packet dropping.

To make ALICE adaptive without introducing negotiation overhead, the authors of [18] proposed ALICE-FP that leverages the Frame Pending (FP) bit made available by the underlying TSCH protocol [4]. When set to 1, the FP flag indicates that the recipient should remain active in the next timeslot and on the same channel, unless the next cell is already allocated for other purposes. Using this very simple mechanism, a node can transmit more data packets than the number of scheduled cells, thus facing temporary traffic peaks.

As a final remark, we observe that the adaption mechanism used in ALICE-FP is similar to the On-demand Provisioning mechanism used in OST. However, it is simpler as it uses a single bit instead of a bitmap. As a consequence, it is less efficient but introduces zero overhead (the FP bit is already present in the Frame Control Field of the TSCH frame).

### E. MINIMAL SCHEDULING FUNCTION (MSF)

As emphasized above, the 6TiSCH *Minimal Scheduling Function* [5] takes a hybrid approach, as it combines autonomous and distributed scheduling and allocates both *autonomous* and *negotiated cells*. Autonomous cells are allocated autonomously (i.e., without neighbor-to-neighbor




(a) Considered RPL topology (b) ALICE unicast slotframe for node B

**FIGURE 3.** An example of ALICE scheduling.

negotiation), using a hash function to compute their timeslot and channel offset, and are used for control information. Instead, negotiated cells are allocated (and deallocated) through the 6P protocol and allow to adapt to time-varying traffic conditions.

Specifically, MSF uses the three following slotframes:

- **Slotframe 0**: used for the bootstrap traffic. It follows the Minimal 6TiSCH Configuration [24] and provides a single shared cell for all kinds of traffic. Data packets are sent in unicast and acknowledged. Since the cell is shared by all the nodes, the collision probability increases with the node density. However, this single shared cell is intended to provide a basic connectivity when no other cells are scheduled.
- **Slotframe 1**: used to schedule autonomous cells.
- **Slotframe 2**: used for cells negotiated through the 6P protocol.

MSF specifications [5] recommend to use the same length for all of the three slotframes, so as to avoid collisions between autonomous and negotiated cells. Following this approach, the collision can be prevented by avoiding using the same cell.

At bootstrap time, each node can use the static cell on Slotframe 0 to join the network. Once the node has selected a preferred parent, two autonomous cells are instantiated in Slotframe 1, namely an *Autonomous Rx Cell* (AutoRxCell) and an *Autonomous Tx Cell* (AutoTxCell). The AutoRxCell is used for receiving control messages and is permanently allocated at joining time, while the AutoTxCell is allocated only after the node has selected its preferred parent and changes when a new parent is chosen. In addition, it is shared with all the other children of the same parent. Finally, it is not permanently scheduled, but added when there is a control message to send, and deleted just after. Both the autonomous cells are allocated by means of a shared hash function.

Negotiated cells are allocated and deallocated dynamically, based on traffic requirements, following a utilization-based approach. Initially, each node negotiates a single cell with its parent node. Then, MSF periodically (every MAX_NUM_CELLS) checks the utilization of the node, defined as the percentage of used cells with respect to scheduled cells. The algorithm considers two thresholds to decide when to add, or remove, a cell. As shown in Algorithm 1, if the cell utilization is higher than the upper threshold (LIM_NUMCELLSUSED_HIGH), one more cell is negotiated with the parent node. Instead, if the utilization falls below the lower threshold (LIM_NUMCELLSUSED_LOW), one cell is deleted from the current schedule.

**Algorithm 1** MSF Algorithm

**Input**:
*NCE* = Number of elapsed negotiated cells
*MAX_NUM_CELLS* = Max number of elapsed negotiated cells
*NCU* = Number of negotiated cells used for transmission
*LIM_NUMCELLSUSED_HIGH* = Threshold to add negotiated cell
*LIM_NUMCELLSUSED_LOW* = Threshold to delete negotiated cell
**Output**:
ADD/DEL one negotiated cell
**if** *NCE* > *MAX_NUMCELLS* **then**
  **if** *NCU* > *LIM_NUMCELLSUSED_HIGH* **then**
    trigger 6P to **ADD** one negotiated cell
  **if** *NCU* < *LIM_NUMCELLSUSED_LOW* **then**
    trigger 6P to **DEL** one negotiated cell

MSF is designed to operate in a wide range of application domains. However, it is optimized for applications with upstream traffic from the nodes to the root [5]. Downward traffic is assumed to be sporadic and its management is not specified. In the implementation used for our experiments, downward traffic is managed through shared cells.

As a final remark, it may be worthwhile pointing out that, according to MSF specifications [5], autonomous cells are used *only for control packets*, while negotiated cells carry data packets. However, this limitation could be removed by allocating more autonomous cells and/or using them also for data packets. Of course, this would reduce the overhead due to 6P negotiations. In our analysis we did not investigate this option.

## IV. EVALUATION METHODOLOGY

To compare the performance of the selected SFs we used simulation and considered a number of different scenarios that are representative of different real-world use cases. We implemented the three selected SFs in the Contiki-NG Operating System (OS), a popular OS for IoT networks that runs on a wide range of hardware platforms. Contiki-NG already supports the 6TiSCH basic operations, which simplifies the implementation of the SFs. Specifically, for our simulations we exploited Cooja [25], a simulator for IoT networks that is part of the Contiki-NG suite. It supports the emulation of hardware components of IoT nodes on which the same code written for real devices can be run. In all the simulation experiments, we used the RPL routing protocol (with default parameters) to allow multi-hop communication.

To compare the performance of the considered adaptive autonomous SFs, we considered the following metrics.

- **Packet Delivery Ratio** (**PDR**), defined as the ratio between the overall number of (data) packets received by the final destination and the number of packets sent by all the nodes in the network. It measures the *end-to-end reliability* provided by the SF.
- **End-to-end Latency**, defined as the time interval between the generation of a (data) packet at the source node and its correct reception at the final destination. This metric measures the *timeliness* of the SF in delivering data.
- **Duty Cycle**, defined as the ratio between the number of cells in which a node remains awake to receive/transmit a data packet and the slotframe size. This metric provides an indication of the amount of communication *resources consumed* by a node when using a given SF, as well as an indirect measurement of its *energy consumption*.
- **Control Overhead**, defined as the number of control bits transmitted, on average, by each node to adapt to traffic conditions. It measures the adaptation overhead introduced by each SF.

In some experiments we also measured additional metrics to provide a further insight in the behavior of the considered SF. These metrics will be introduced below, when used.

In our analysis, we considered four different scenarios, characterized by different communication paradigms (*many-to-one*, *one-to-many*, *many-to-many*) and traffic patterns (*periodic* or *bursty*), so as to investigate different IoT use cases. In the first two scenarios, we considered the case where all the nodes send their data to a collection point, i.e., the root node (*many-to-one communication*). They both refer to data collection applications (e.g., monitoring applications), but differ in the traffic generation pattern. In the first scenario nodes report data *periodically*, as in environmental monitoring applications. In the second scenario, the traffic pattern is *bursty*, i.e., nodes occasionally send a burst of packets, while for the rest of the time they remain idle. This scenario is representative of event-driven applications (e.g., an alert system), in which the detection of a certain event triggers the generation of a number of packets (e.g., an image taken from a camera). In the third scenario we consider the case where the root sends packets to all (or many) nodes in the network (*one-to-many communication*). This happens, for instance, when the root needs to send commands/updates to sensor or actuator nodes. Finally, in the last scenario we consider *one-to-one communication*, where a device (e.g., a sensor) sends a flow of data packets to another device (e.g., an actuator). This occurs in all those applications (e.g. industrial instrumentation) that require a direct communication between devices, i.e. machine-to-machine (M2M) communication.

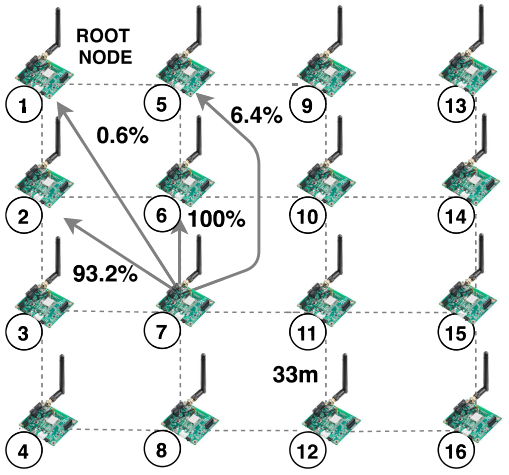



**FIGURE 4.** Grid topology with average packet delivery probabilities.

Simulation experiments were run for a fixed period of time, namely, 1 hour. In order to obtain statistically sound results, we performed 10 independent replicas for each simulation experiment. For each metric, we computed the confidence interval on all the replicas, with a 95% confidence level. In some cases, the confidence interval is very small and difficult to appreciate in the plots shown below.

## V. PERFORMANCE ANALYSIS

In this Section, we present the results obtained in our study. In our experiments, we considered the network depicted in Figure 4, namely a grid topology with a number of nodes ($N$) distant 33m from each other. To simulate the unreliability of wireless communications, each link was modeled through the *Multi-path Ray-tracer Medium* (MRM) model [25]. MRM implements ray-tracing techniques with various propagation effects (e.g., multi-path, refraction, diffraction, etc.) and associates a *Packet Delivery Probability* (PDP) to each link. Obviously, the PDP of a link depends on the distance between the connected nodes. Also, it changes over time, due to propagation effects and concurrent transmissions. Figure 4 shows, for each link, the associated average PDP value. Each node is configured to send UDP data packets with a certain generation period ($P$).

Table 1 shows the parameter values used in our experiments. For general parameters, we considered the same values used in previous simulation analysis [18], [23]. For

**TABLE 1. Parameter settings.**

| Parameter | Value |
|---|---|
| MACMinBE | **1** |
| MACMaxBE | **5** |
| MACMaxFrameRetries | **7** |
| HOPPING_SEQUENCE_EBs | **4 channels** |
| HOPPING_SEQUENCE_DATA | **16 channels** |
| Queue Buffer | **16** |
| TSCH timeslot duration | **10ms** |
| 6top Timeout | **32s** |
| RPL Mode | **Storing** |
| RPL Objective Function | **MRHOF - ETX** |
| RPL DIO interval doublings | **8** |
| RPL redundancy threshold | **10** |

parameters characterizing the three SFs, we used the same values suggested by the authors, whenever possible. To make the comparison fair, we considered all the 16 available channels for all the SFs. Finally, since the performance of ALICE-FP is strongly influenced by the slotframe size (while both MSF and OST are not sensitive to this parameter), we performed a set of preliminary experiments to select the slotframe size in the scenarios under consideration. Specifically, we ran a number of experiments in the scenarios analyzed in the paper, with different slotframe sizes (we considered values suggested by the standard). We observed that the best performance is obtained by ALICE-FP when the slotframe size is equal to 29. Hence, below we will use this value.

In our study, we compared the three considered SFs, both in terms of performance (end-to-end reliability and delay), resource consumption (duty cycle) and complexity (control bits), with respect to different factors, such as number of IoT nodes and traffic rate. In the following, we first discuss the results obtained in the scenarios characterized by Many-to-One communication, with both periodic traffic (Section V-A) and bursty traffic (Section V-B). Then, we will analyze the scenarios with One-to-Many communication (Section V-C) and One-to-One communication (Section V-D). The complexity of the considered SFs will be discussed in Section VI.

### *A. MANY-TO-ONE COMMUNICATION WITH PERIODIC TRAFFIC*

In this scenario, each IoT node generates an UDP packet of fixed size (60 bytes), destined to the root node, with a constant period $P$. We start our performance comparison by analyzing the behavior of the considered SFs with respect to the number of IoT nodes in the network. To this end, we considered an increasing number of nodes, in the range [16-64]. In this first set of experiments, the packet generation period ($P$) was constant and equal to 10s [26]. In a second set of experiments, described later, we will investigate the behavior of the same SFs, with respect to the traffic rate, by changing the $P$ value in a range typical for industrial applications [26].

Figure 5 shows the Packet Delivery Ratio (PDR), end-to-end delay, and average duty cycle for the three considered SFs, with different networks sizes. The average duty cycle is computed on all the nodes, and indicates the percentage of time during which nodes remain active for transmitting or receiving data packets. Hence, as mentioned above, this metric is an indirect measurement of energy efficiency. Also, it provides an indication of the amount of bandwidth consumed by a given SF.

In terms of end-to-end reliability (Figure 5-a), in this scenario, all the considered SFs perform quite well, as the PDR is always very close to 100%. As expected, the performance of ALICE-FP decreases slightly when the number of IoT nodes increases over a certain threshold. This is because, with a high number of nodes, it may happen that some links have to manage more than one packet per slotframe. Since the adaptation mechanism in ALICE-FP is very simple, it is unable to provide a significant amount of additional bandwidth, resulting in increased delay (see also Figure 5-b) and a small fraction of dropped packets.

In terms of resource consumption, OST is the most efficient option, as it exhibits the lowest average duty cycle (Figure 5-c). This can be easily justified by observing that OST allocates the minimum number of cells required to manage the traffic on the link, while ALICE-FP allocates a fixed amount of bandwidth, irrespective of the traffic load on the link. MSF has the highest average duty cycle, as the latter also considers, in addition to cells used for data packets, also cells used by the 6P protocol, i.e., the negotiation overhead that is not present in OST and ALICE-FP.

The minimum average duty cycle in OST comes at the cost of an increased end-to-end delay experienced by data packets. As observed, OST tries to allocate the minimum number of cells required to manage the traffic on the link, while MSF and, above all, ALICE-FP tend to overprovision nodes. Hence, typically OST exhibits the largest end-to-end delay and ALICE-FP the lowest one.

Since both OST and ALICE-FP leverage different allocation mechanisms, it may be interesting to analyze the contribution of each such allocation mechanism. To this end, we measured the percentage of data packets transmitted in the different kinds of cells. The results obtained, are shown in Figure 6, for both OST (a) and ALICE-FP (b). As a general remark, we can observe that, in this scenario, the percentage of packets transmitted through on-demand cells (FP cells with ALICE-FP) is approximately the same in OST and ALICE-FP. Focusing on OST, we can observe that, AUS cells are used very rarely (as expected). Instead, the percentage of packets transmitted in cells allocated through On-demand Provisioning is surprisingly high (about 50% with 64 IoT nodes). We would expect a lower percentage, since the traffic pattern is Periodic and, hence, the Periodic Provisioning mechanism should adapt the slotframe size to the link traffic rate.

We investigated this point and found that the observed result is mainly due to the allocation policy used by the

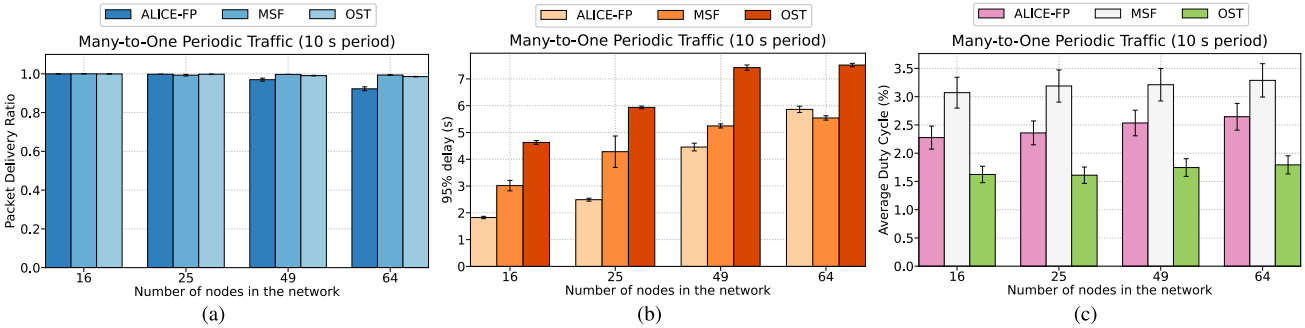



**FIGURE 5.** (a) Packet delivery ratio, (b) End-to-end delay, and (c) Duty cycle for increasing number of IoT nodes. ***Many-to-One*** communication, ***Periodic*** traffic, $P = 10s$.

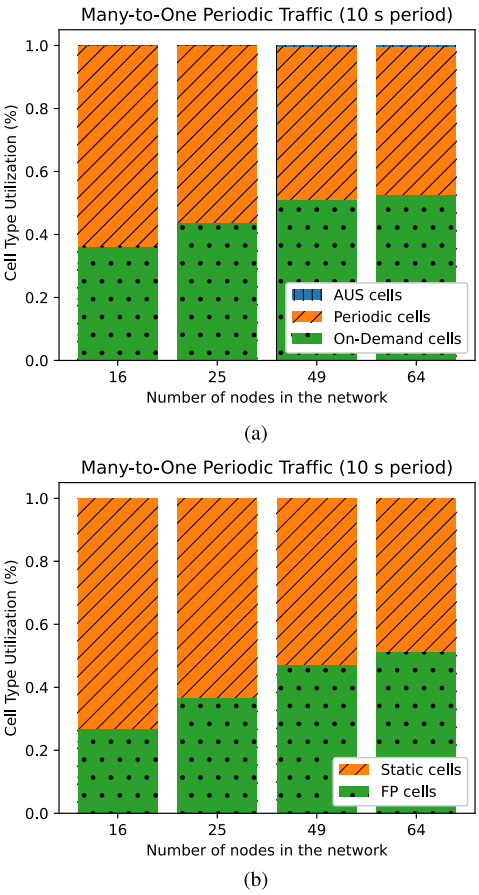



**FIGURE 6.** Contribution of the different allocation mechanisms in OST (a) and ALICE-FP (b).

Periodic Provisioning mechanism (see Section III-C for details). It may be worthwhile recalling here that, to derive the appropriate slotframe size for a certain link, the sender node estimates the *Inter-Packet Slots* (*IPS*), and then selects the slotframe size as $2^N$, such that $2^N \leq IPS < 2^{N+1}$. However, if no time offset is available at the receiver side for the proposed slotframe, the sender is forced to double the slotframe size ($2^{N+1}$), which results in *under-provisioning*. Now, if the number of allocated cells is lower than the number of generated packets, packets will accumulate in the local buffer, and the On-demand Provisioning mechanism will activate very often. Obviously, the more the nodes are, the higher the probability is of having no time offset available at the receiver. This explains why the percentage of packets using on-demand cells increases as the number of nodes in the network grows up.

So far, we have analyzed the performance of the considered SFs with an increasing number of IoT nodes. In order to assess the impact of the traffic rate, we carried out an additional set of simulation experiments, where we varied the packet generation period $P$, while keeping the number of nodes constant. Specifically, we considered two network sizes, namely 25 and 64 IoT nodes.

Figure 7 shows the considered metrics vs. the Packet Generation period (i.e., traffic rate) in a network of 25 nodes. We can see that, in such a network, the traffic rate has no significant impact on the performance of the three considered SFs. The delivery ratio is almost the same for all of them and very close to 100%, except when the traffic rate is very high (i.e., 1 packet/sec), when ALICE-FP degrades and MSF tends to slightly outperform OST. The impact of the traffic rate on the average duty cycle is also very light, and OST confirms to be the most efficient option, in terms of resource consumption, followed by ALICE-FP and MSF. Finally, in terms of end-to-end delay, ALICE-FP behaves in a very different way, with respect to both MSF and OST.

Specifically, the end-to-end delay of ALICE-FP does not depend on the traffic rate, mainly due to the static allocation that typically results in overprovisioning. Only when the traffic rate is high, there is an increase in the delay because the traffic load at more loaded links exceeds the allocated bandwidth and the FP bit is not able to provide the required extra bandwidth. Instead, for both MSF and OST, the end-to-end delay tends to increase when the packet generation period increases (i.e., the traffic rate decreases). This is because they tend to allocate fewer resources when the traffic rate is light and, hence, packets tend to accumulate in the local buffer. In particular, OST introduces the highest delay because its Periodic Provisioning mechanism adjusts the slotframe size, depending on the packet generation period, in order to minimize the number of allocated cells. The step-wise behavior of the end-to-end delay in OST is due to the inter-dependence

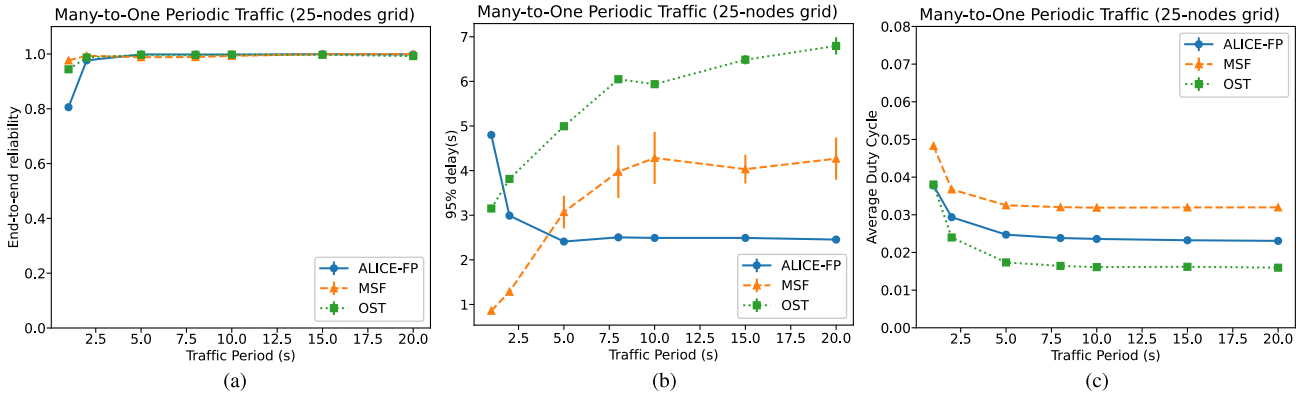



**FIGURE 7.** (a) Packet delivery ratio, (b) End-to-end delay, and (c) Duty cycle for increasing packet generation period (*P*). ***Many-to-one*** communication, ***Periodic*** traffic, $N = 25$ nodes.

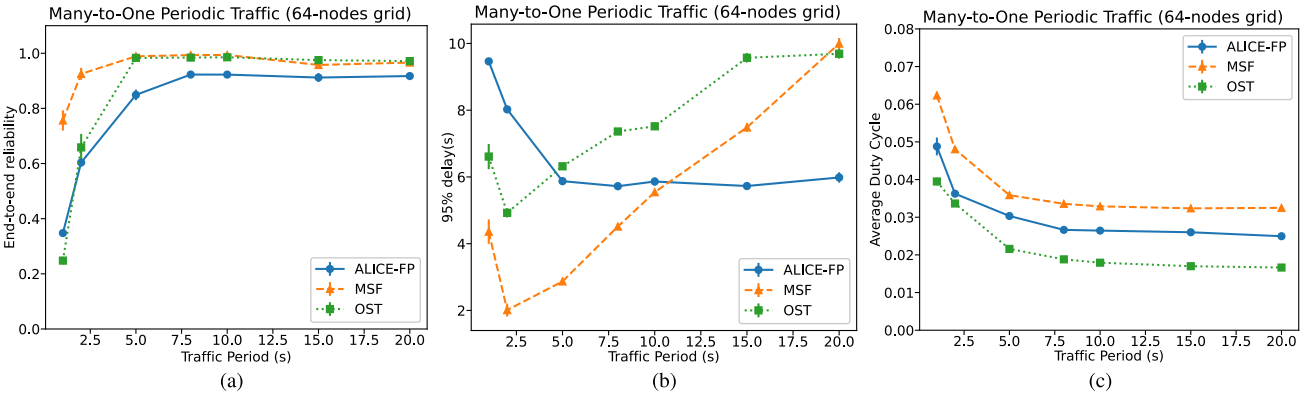



**FIGURE 8.** (a) Packet delivery ratio, (b) End-to-end delay, and (c) Duty cycle for increasing packet generation period (*P*). ***Many-to-one*** communication, ***Periodic*** traffic, $N = 64$ nodes.

between the packet generation period and the slotframe size used by the Periodic Provisioning mechanism.

Also Figure 8 analyzes the impact of the traffic rate on the performance of the considered SFs, however, it refers to a larger network size (64 nodes). The general trend is the same as that observed in Figure 7, with some significant differences, which are discussed in the following. In terms of end-to-end reliability, we observe that ALICE-FP is not able to provide 100% delivery ratio, even when the traffic rate is light (this behavior was already observed and explained, with reference to Figure 5), while both OST and MSF provide almost full reliability, unless the traffic rate is high (i.e., $P < 5s$). When the traffic rate is high, all the SFs exhibit a significant drop in the delivery ratio, however, MSF significantly outperforms both OST and ALICE-FP. The delivery ratio provided by OST, in such conditions is similar to (if not lower than) that provided by ALICE-FP. We found that this behavior is mainly due to the instability of the RPL protocol, which results in frequent parent changes. These parent changes, however, are much more frequent with OST and ALICE-FP, as shown in Figure 9-a (which refers to the case when $P = 1s$). We can observe, that the number of parent changes, with OST and ALICE-FP, increases considerably with the network size. Hence, the performance of OST and ALICE-FP is affected significantly by the RPL protocol, when traffic rate is high. This can be explained considering the limited number of shared timeslots allocated by ALICE-FP and OST for the transmission of control messages, including RPL messages. While MSF allocates a certain number of shared timeslots for control message transmissions, as it is needed for 6P messages, ALICE-FP and OST allocate only a few shared timeslots. Specifically, ALICE-FP and OST statically allocate a single cell in a dedicated slotframe for the management of broadcast control messages. The difference between ALICE-FP and OST, instead, can be explained with the different approach used to handle unicast RPL messages. While ALICE-FP transmits such messages by exploiting the same cells used for data traffic, OST uses the AUS slotframe. The latter, following the node-based scheduling of Orchestra-RB, does not allow a management of unicast control messages as efficient as that of ALICE-FP. The reduced number of transmission opportunities results in a less efficient circulation of RPL messages (delayed messages plus collisions), thus resulting in more frequent preferred parent changes.

### B. MANY-TO-ONE COMMUNICATION WITH BURSTY TRAFFIC

In this Section, we still consider many-to-one communication (i.e., upward traffic), but assume that the traffic pattern is *bursty*, as in the case of event-driven monitoring applications.

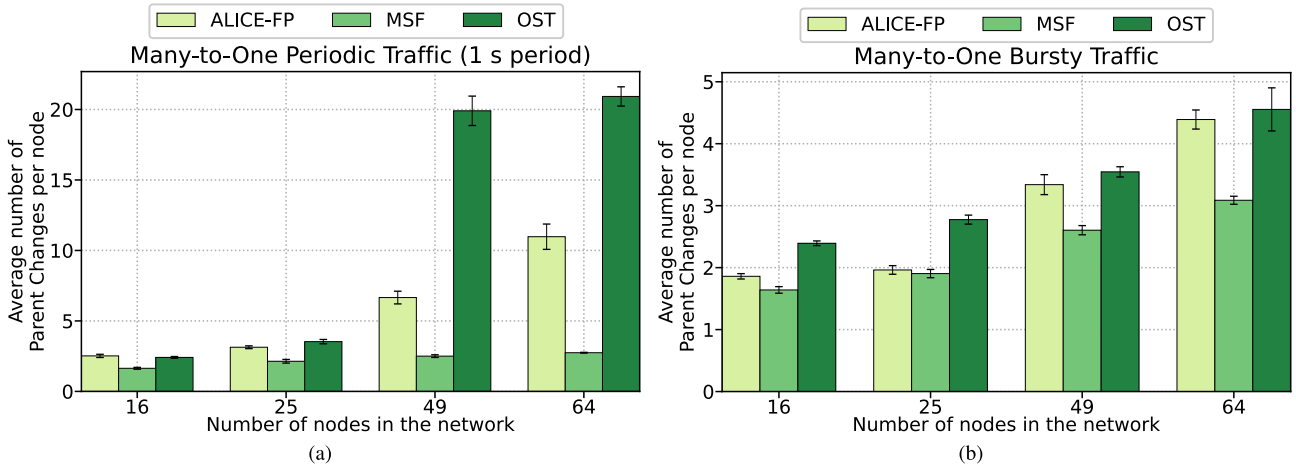



**FIGURE 9.** Average number of parent changes, for different network sizes. ***Many-to-one*** communication. ***Periodic*** traffic, $P = 1s$ (a); ***Bursty*** traffic (b).

The goal is to assess the performance of the considered SFs in a case where IoT nodes alternate between inactive periods and periods characterized by high traffic intensity. More specifically, each node alternates between OFF periods, when no packets are generated, and ON periods, during which 100 UDP packets, of fixed size (60 bytes), are generated (with destination the root node) with a period of 0.5 seconds. After sending the last UDP packet the node stops, until the beginning of the next ON period. The duration of OFF periods is a random variable with uniform distribution between 1 and 20 minutes. In all the experiments, we considered a network with increasing size, from 16 to 64 IoT nodes.

Figure 10 shows the performance of the considered SFs, with an increasing number of IoT nodes, in the scenario with bursty traffic. As above, OST has the lowest average duty cycle, since it tries to minimize the amount of allocated resources. This comes at the cost of an increased latency and, in fact, OST experiences the highest end-to-end delay. From the analysis of this scenario it also emerges that, when the traffic is bursty, OST is not able to provide full reliability, even when the network is small (e.g., with 25 nodes). As shown in Figure 10-a, OST exhibits the lowest packet delivery ratio, even below ALICE-FP. This is because the Periodic Provisioning mechanism takes some time to adapt to the new conditions when a node passes from the OFF state (no transmissions) to the ON state (periodic transmissions with $P = 0.5s$), and the On-demand Provisioning is of limited help in such conditions. Instead, ALICE-FP can leverage the basic amount of bandwidth, provided by the static allocation mechanism, increased through the additional cells provided by the FP mechanism in a rapid manner. Finally, MSF provides the highest delivery ratio, as it is able to adapt more quickly than the others to traffic variations. However, it uses much more cells, also due to 6P negotiation messages, as shown by the highest average duty cycle, and introduces a delay larger than ALICE-FP. Figure 9-b shows that, even in this scenario, OST and ALICE-FP experience more parent changes than MSF. However, now the difference is not so apparent as before. Overall, in the bursty traffic scenario, ALICE-FP is the best option.

### C. ONE-TO-MANY COMMUNICATION

In the previous sections we have analyzed the considered SFs when the traffic is directed *upward*, i.e., from IoT nodes to the root. In this Section we want to investigate the case when the traffic flows *downward*, i.e., from the root to nodes. This is not so frequent in IoT environments. However, it happens when the root needs to send a command to sensors (e.g., updated parameter values, configuration changes, etc.) or actuators (e.g., actions to be performed). In our experiments, we assumed that the root sends UDP packets of fixed size (60 bytes) in the network, with a certain period $P$. As above, we considered a network with increasing number of IoT nodes (from 16 to 64).

In the first set of experiments, we simulated the case of *sporadic downward* traffic, which is the typical case of downward traffic in IoT networks. We assumed that the packet generation period is very large (i.e., 1 minute) and each packet is directed to a single destination, randomly selected among all nodes. Under such conditions, we observed that all the considered SFs perform reasonably well, irrespective of the network size. Specifically, in the worst case (i.e., with N=64), we observed a PDR above 95% for all the considered SFs, and an end-to-delay of about 2s and 4s for ALICE-FP and OST, respectively. MSF introduces a higher, yet acceptable, delay (about 13s). The reasons for this larger delay for MSF will become clear with the second set of experiments.

In the second set of experiments, we increased significantly the downward traffic injected into the network, in order to investigate how the considered SFs behave in high workload conditions. To this end, we decreased the packet generation period ($P$) at the root and assumed that each packet is sent to a certain number of destinations. Figure 12 shows the performance of the three considered SFs, for different network

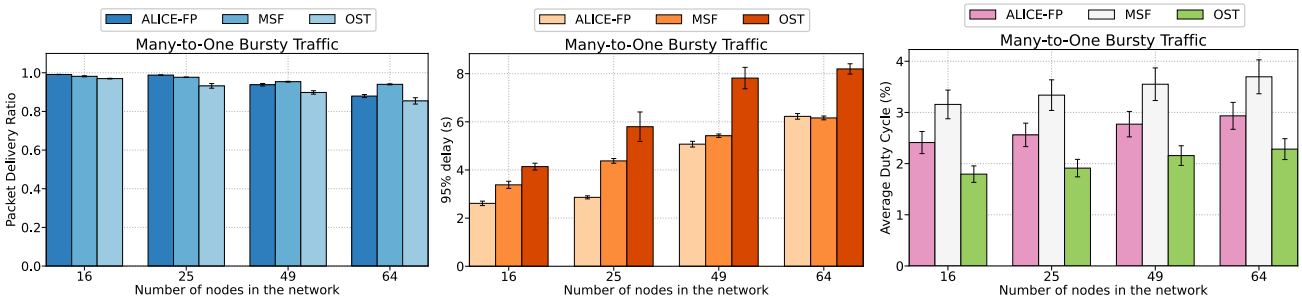



**FIGURE 10.** (a) Packet delivery ratio, (b) End-to-end delay, and (c) Duty cycle for increasing number of IoT nodes. ***Many-to-one*** communication, ***bursty*** traffic.

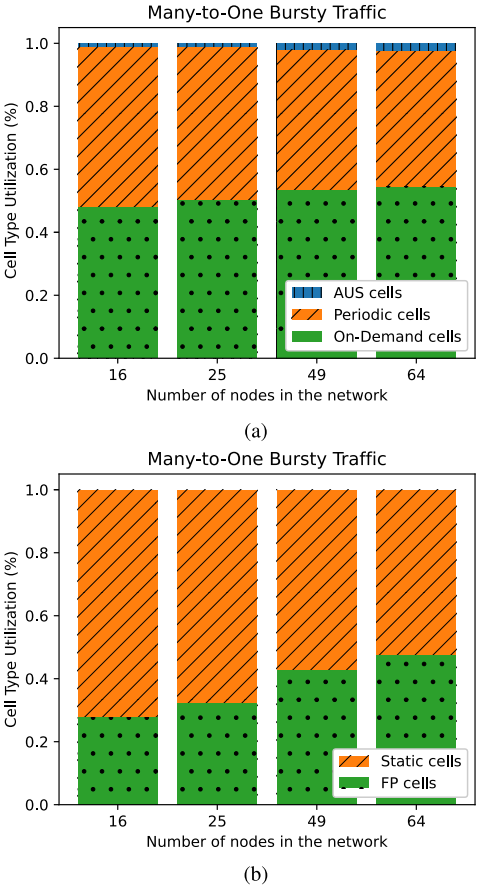



**FIGURE 11.** Contribution of the different allocation mechanisms, with ***Bursty*** traffic, in OST (a) and ALICE-FP (b).

sizes, when $P = 30s$ and a copy of each packet is sent by the root node to all the nodes in the network. This corresponds to a very high aggregate downward traffic.

We can observe that, in this specific scenario, MSF exhibits very poor performance as the delivery ratio is very low, even with small network sizes. This behavior can be explained by recalling that MSF is optimized for upward traffic. Specifically, cells are allocated only for communication from the children to the parent, while communication in the reverse direction is managed through shared cells. Instead, both ALICE-FP and OST allocate cells in both directions and this explains their better performance in this scenario.

OST and ALICE-FP have similar performance, in terms of delivery ratio. As above, OST provides a lower (average) duty cycle and a larger end-to-end delay. Comparing the results in Figure 12 with those obtained with upstream periodic traffic (Figure 5), we can observe that the end-to-end delay with downstream traffic is significantly lower than that with upstream traffic, for both OST and ALICE-FP (we verified that this behavior also holds when the packet generation period is 30s in both cases). The different values of the end-to-end delay, with upward and downward traffic, can be explained as follows. When the traffic flows upward, all the children of the same parent must compete for using the resources of their parent. Instead, when the traffic is downward, each node receives packets only from its parent, without competing with other nodes, and this results in a lower waiting time.

In conclusion, both OST and ALICE-FP perform quite well in this scenario. The former minimizes energy/resource consumption at the cost of increased end-to-delay, while the latter takes the opposite approach.

### D. ONE-TO-ONE COMMUNICATION

In this final scenario, we consider the case of device-to-device communication. This is a very common situation in IIoT environments, where the source device may be a sensor sending a data flow to an actuator. In this scenario, the path followed by a packet includes both an upward component and a downward component. A packet sent by a source device is forwarded upward, along the RPL DODAG, until it reaches the root of the subtree including the destination node. Then, it is forwarded downward to the destination node, again following the DODAG.

In our experiments, we assumed four device-to-device simultaneous communications, with source and destination nodes located at opposite locations in the grid, so that packets have to travel upward from the source to the root and, then, downward from the root to the destination. For instance, with reference to the $4 \times 4$ grid shown in Figure 4, nodes 1, 2, 3,

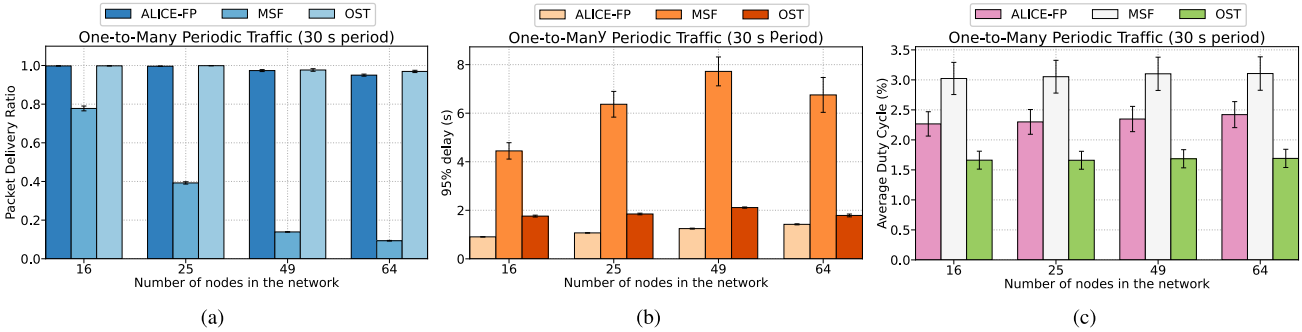



**FIGURE 12.** (a) Packet delivery ratio, (b) End-to-end delay, and (c) Duty cycle for increasing number of IoT nodes. *One-to-many* communication, *periodic* traffic, $P = 30s$.

and 4 are the source nodes, while nodes 13, 14, 15 and 16 are the corresponding destinations. Each source node generates a periodic flow of UDP packets, of fixed size (60 bytes), with a period $P$ of 30s. As in the previous scenarios, we considered increasing network sizes (from 16 to 64).

Figure 13 shows the performance of the three SFs in this scenario, which is a mix of the previously considered scenarios, as it includes both upstream and downstream traffic. As expected, MSF performs quite badly, also in this scenario, due to the downward traffic component. OST and ALICE-FP exhibit a similar delivery ratio, with OST slightly outperforming ALICE-FP, when the network size (and, consequently, the aggregate workload) tends to increase. Moreover, as above, OST provides a lower (average) duty cycle, at the cost of an increased end-to-end delay, while ALICE-FP takes the opposite approach and favors the timeliness in packet delivery, at the cost of a higher resource consumption.

## VI. COMPLEXITY ANALYSIS

In the previous Section we have evaluated the three considered SFs in terms of performance (end-to-end reliability and delay) and resource consumption (duty cycle). To give a complete picture of the situation, it is also important to look at the complexity of each SF. A quantitative evaluation of the complexity may not be an easy task, as the three SFs take different approaches, and also because complexity may be regarded from different points of view (i.e., computation complexity or communication overhead).

A quantitative comparison of the *computation complexity* associated with the three SFs is almost impossible to perform, since the considered SFs perform different actions, whose computational cost is very difficult to evaluate and compare. Hence, we just provide a qualitative evaluation, based on the analysis of the actions performed by the three SFs. ALICE-FP has the lowest complexity, as a node just sets the FP bit in the underlying TSCH frame, whenever there are queued packets. In OST, in case of queued packets, a node computes the *Subsequent Timeslot Schedule* (STS) and piggybacks it on the packet destined to B. In addition, each node continuously estimates the optimal PTS/PRS size, for each link it is involved in, and piggybacks the new size in the packet. On the other side, the receiver node must check the availability of a time offset for the requested slotframe size and notify it to the sender. Otherwise, a negotiation is started. Finally, in MSF, each node periodically computes the cell utilization and, depending on its value, decides to allocate or deallocate one cell, through a 6P transaction. The 6P protocol is also part of the complexity associated with MSF. Based on the above analysis, we can conclude that ALICE-FP has an almost-zero computation complexity, while the complexity of both OST and MSF is non-negligible.

The *communication overhead* (or control overhead) of the three SFs can be quantified in terms of number of additional control information transmitted, by each node, for adapting to changing traffic conditions. To this end, we need to consider the number (and type) of adaption actions performed by each node when using a specific SF, and the associated cost, in terms of additional number of control bits transmitted by the node. For the sake of space, in our (complexity) analysis, we only consider the case of Many-to-One communication with Periodic ($P = 10s$) and Bursty traffic. However, following the same approach, the analysis can be easily extended to the other scenarios.

Figure 14 shows the average number of adaptation actions, per second, performed by each node. The actions performed by a node depend on the particular SF, as listed below:

- **ALICE-FP**: request of an on-demand cell (FP mechanism);
- **OST-PP**: request to set up a new PTS/PRS size (Periodic Provisioning);
- **OST-ODP**: request of an on-demand cell (On-Demand Provisioning);
- **MSF**: request to allocate/deallocate one cell (6P transaction).

From the analysis of Figure 14, we can observe that MSF performs a very low number of 6P transactions, basically those required initially for allocating the appropriate number of cells. This is because the traffic is periodic and the utilization-based approach used in MSF avoids frequent

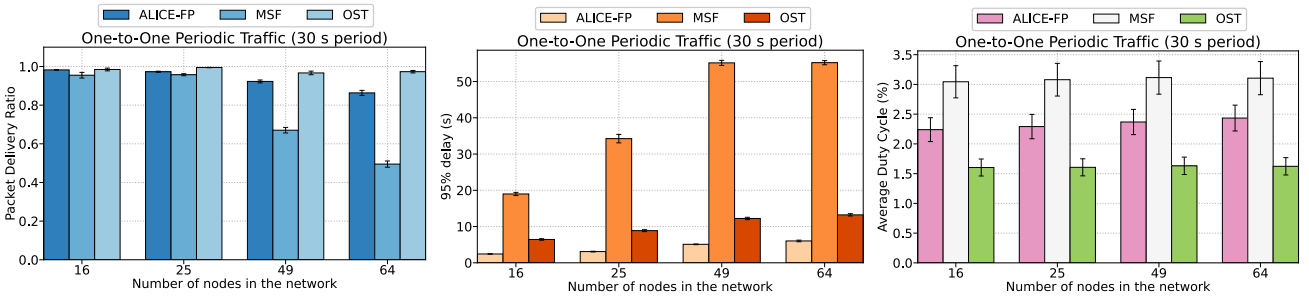



**FIGURE 13.** (a) Packet delivery ratio, (b) End-to-end delay, and (c) Duty cycle for increasing number of IoT nodes. ***One-to-one*** communication, ***periodic*** traffic, $P = 30s$.

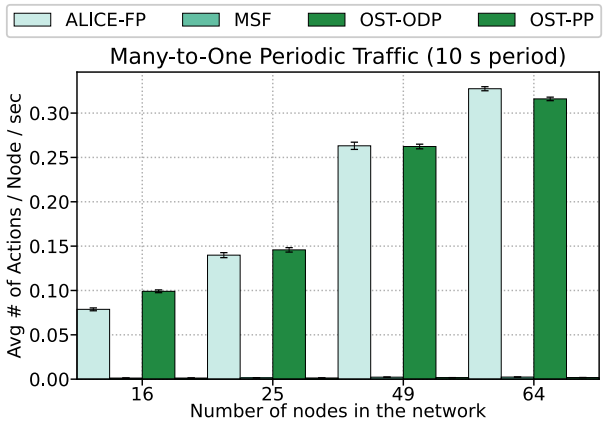



**FIGURE 14.** Average number of adaptation actions performed by each node in a second, ***Many-to-one*** communication, ***periodic*** traffic, $P = 10s$.

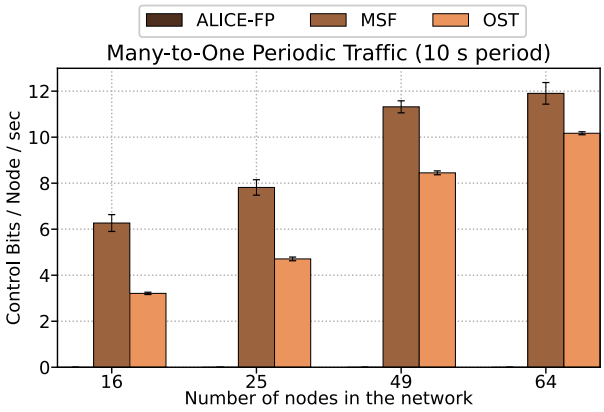



**FIGURE 15.** Average number of control bits transmitted by each node in a second, ***Many-to-one*** communication, ***periodic*** traffic. $P = 10s$.

allocations and deallocations (at the cost of overprovisioning). Similarly, the number of slotframe adjustments in OST is extremely limited. Instead, the On-demand Provisioning mechanism is activated very frequently, and the same also occurs for the FP mechanism in ALICE-FP (these results are in accordance with those shown in Figure 6). Overall, the frequency of adaptation actions performed by OST is similar to that of ALICE-FP, and even larger in case of low number of nodes.

The rate of actions performed by each node does not provide a clear measurement of the control overhead introduced by each SF, as different actions require a different number of control bits to be implemented. Hence, we need to consider the number of control bits transmitted for each of the above-mentioned actions. Specifically, for ALICE-FP there is no additional control information transmitted, as the *Frame Pending* bit is always present in the TSCH frame and must be set appropriately at each transmission. In OST, the transmission of the *Subsequent Timeslot Schedule* requires 16 bits (and 16 additional bits for the response). Similarly, the request of a new PTS/PRS *size* asks for 16 bits for the request, and 16 more bits for the acknowledgment (assuming that there is no negotiation). Finally, for MSF, the request of a cell allocation/deallocation requires the transmission of two 6P messages, namely a *6P Request* and the corresponding *6P Response*. Since 6P messages are transmitted as separate control messages, the control overhead corresponds to two TSCH cells, under the optimistic assumption that the Request is always successful. Assuming that each cell has a duration of 10ms, the corresponding number of bits, per cell, is 2500 (assuming a bit rate of 250 Kbps).

Based on these remarks and leveraging the results in Figure 14, Figure 15 shows the number of control bits per second transmitted, on average, by each node, when using the three considered SFs. As expected, MSF introduces the highest control overhead, as it uses separate 6P messages for negotiation. The total control overhead introduced by OST is lower than MSF, but significant, due to frequent activations of the On-demand Provisioning mechanism. Finally, as anticipated, ALICE-FP has zero control overhead.

Finally, we also measured the control overhead when the traffic is bursty (still considering Many-to-One communication). Figure 16 and 17 show the average number of actions per second and the associated control overhead, for the considered SFs. As a general comment, we can observe that

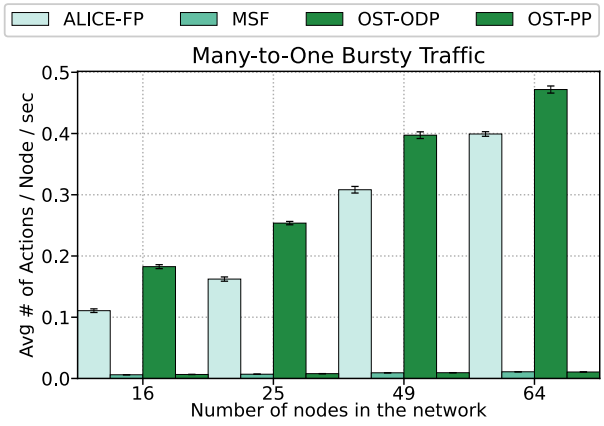



**FIGURE 16.** Average number of adaptation actions performed by each node in a second, *Many-to-one* communication, *bursty* traffic.

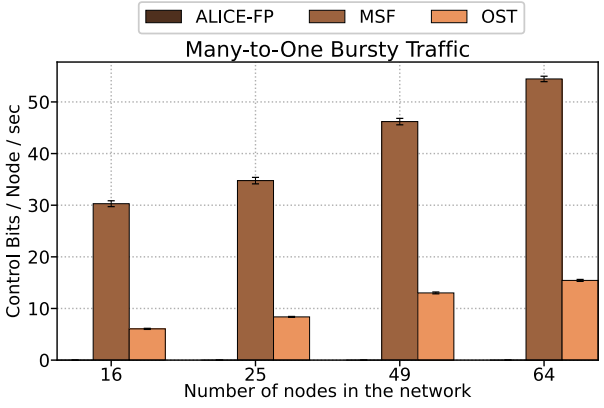



**FIGURE 17.** Average number of control bits transmitted by each node in a second, *Many-to-one* communication, *bursty* traffic.

all the SFs perform more adaptation actions than before (i.e., with Periodic traffic), as this scenario is more dynamic. In terms of control bits per second, now MSF predominates over OST, due to high communication overhead associated with 6P transactions. As above, ALICE-FP has zero control overhead.

## VII. LESSONS LEARNED

In this Section, we summarize the lessons we have learned, through our analysis, about the behavior of the considered SFs and present a set of guidelines for helping the designer of IoT-based systems in selecting the most appropriate SF, depending on the specific use case.

As a general remark, it may be worthwhile observing that OST and ALICE-FP are both adaptive autonomous SFs, but they take a different approach in cell allocation. ALICE-FP is basically static and leverages a simple adaptation mechanism to manage light traffic changes, while OST is fully dynamic and tries to allocate the minimum number of cells to satisfy the reliability requirements. Hence, ALICE-FP is less complex, more stable, and provides a shorter end-to-end delay; however, it consumes more resources (i.e., bandwidth and energy). Instead, OST minimizes the resource consumption at the cost of increased end-to-end delay. Based on these results, as a general guideline, ALICE-FP may be more appealing for (soft) real-time applications, where guaranteed bandwidth, timeliness, stability of operations, and low complexity are typically key requirements, while a larger resource consumption is the unavoidable price to pay for those. On the other hand, OST appears to be more suited for less critical applications, where timeliness is not the main issue, while energy efficiency is important.

In scenarios characterized by many-to-one (i.e., **upward**) communication with **periodic traffic**, as in monitoring applications with periodic data reporting, the behavior of the SFs is influenced by the aggregate workload on the network. When the aggregate workload is low-to-moderate, all the considered SFs exhibit similar performance and, hence, ALICE-FP is the preferred option, especially in industrial use cases, due to its low complexity, zero control overhead, and long-term stability. It may be worthwhile emphasizing that real-world deployments are typically characterized by low-to-moderate workload. Moreover, in ALICE-FP the static allocation can be adjusted by setting appropriately the slotframe size. This allows to allocate more or less bandwidth at design time.

When the aggregate workload is high (e.g., due to large number of nodes and/or high traffic rate), ALICE-FP is not the best option, because the traffic on more loaded links typically exceeds the available (static) bandwidth and the FP mechanism cannot help so much. However, in these conditions, also OST performs worse than MSF, in terms of end-to-end reliability, mainly due to the high number of parent changes generated by the RPL protocol. Hence, among the considered SFs, MSF is the best option in this scenario. However, it is based on 6P negotiation and, in a previous paper [23], it was shown that E-OTF, another 6P-based algorithm, performs even better than MSF in high workload conditions.

When the traffic conditions are very dynamic, as in scenarios with (**upward**) **bursty traffic**, OST does not work very well, because the Periodic Provisioning mechanism is slow and takes some time to adapt when the traffic conditions change, while the On-demand Provisioning is of limited help in such a dynamic scenario. Among the considered SFs, MSF has the best performance, especially in terms of end-to-end reliability. However, it consumes much more resources than ALICE-FP, due to the negotiation mechanism based on the 6P protocol. Hence, ALICE-FP can be a good compromise also in this scenario.

In scenarios characterized by One-to-Many communication (i.e., **downward traffic**), MSF exhibits very poor performance, because it is optimized for upstream traffic. OST and ALICE-FP have similar performance, in terms of delivery ratio, but ALICE-FP provides a better timeliness at the cost of a larger resource consumption, while OST takes the opposite approach.

Finally, similar conclusions also hold for the One-to-One scenario characterized by a mix of **upward and downward traffic**, as in device-to-device communications.

## VIII. CONCLUSIONS

In this paper, we have focused on adaptive autonomous scheduling for 6TiSCH networks. Specifically, we have compared three SFs that take different approaches in allocating cells for communication and adapting to traffic conditions, namely OST, MSF, and ALICE-FP. We have analyzed the above-mentioned SFs, in four different scenarios, representatives of many real-world use cases, in terms of performance, resource consumption, and complexity.

From our study, it emerges that no SF outperforms the other ones in all the considered scenarios. Instead, different SFs exhibit pros and cons under different conditions. Therefore, in Section VII, we have provided a set of guidelines to select the most appropriate (autonomous) SFs, depending on the specific scenario and operating conditions.

Intuitively, one would expect OST to outperform ALICE-FP, as the former takes a fully adaptive approach, while ALICE-FP has only a limited adaptation capability. In fact, we observed that OST tends to minimize the resource consumption, at the cost of a larger end-to-end delay, while ALICE-FP takes the opposite approach. However, OST is much more complex than ALICE-FP, not only from a computational point of view, but also in terms of additional number of control bits transmitted. Indeed, ALICE-FP provides adaptation with zero overhead, according to the philosophy of autonomous scheduling. Instead, OST has a non-negligible control overhead. In addition, ALICE-FP relies on a basic static allocation. This, typically, results in a non-optimal resource utilization, but provides long-term stability, robustness, guaranteed bandwidth, and timeliness, which are key requirements in industrial settings. Based on these remarks, we believe that ALICE-FP is a very good candidate for real-world industrial IoT applications.

Our comparative analysis is completely based on simulation. We used simulation as it allows to analyze a large number of scenarios and to investigate the impact on performance of many different factors (such as network size, traffic pattern, traffic rate). This is very difficult, if not impossible, to manage through real experiments. At the same time, we are aware that simulation may not capture all the details of the real-world. Therefore, as a future work, we plan to carry out some experiments on a real IIoT testbed available at our CrossLab for Industry 4.0.

## ACKNOWLEDGMENT

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

## REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[2] P. Thubert, *An Architecture for IPv6 Over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)*, document RFC 9030, May 2021.

[3] X. Vilajosana, T. Watteyne, T. Chang, M. Vucinic, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 595–615, 1st Quart., 2020.

[4] *IEEE Standard for Low-Rate Wireless Networks*, Standard 802.15.4-2015 Std 802.15.4-2011, 2016.

[5] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and A. R. Dujovne, *6TiSCH Minimal Scheduling Function (MSF)*, document RFC 9033, May 2021.

[6] P. Thubert, M. R. Palattella, and T. Engel, "6TiSCH centralized scheduling: When SDN meet IoT," in *Proc. IEEE Conf. Standards for Commun. Netw. (CSCN)*, Oct. 2015, pp. 42–47.

[7] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[8] O. Harms and O. Landsiedel, "MASTER: Long-term stable routing and scheduling in low-power wireless networks," in *Proc. 16th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2020, pp. 86–94.

[9] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks," *IEEE Sensors J.*, vol. 16, no. 2, pp. 550–560, Jan. 2016.

[10] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "An evaluation of the 6TiSCH distributed resource management mode," *ACM Trans. Internet Things*, vol. 1, no. 4, pp. 1–31, Oct. 2020.

[11] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. 13th ACM Conf. Embedded Networked Sensor Syst.*, New York, NY, USA, Nov. 2015, pp. 337–350.

[12] S. Kim, H.-S. Kim, and C. Kim, "ALICE: Autonomous link-based cell scheduling for TSCH," in *Proc. 18th Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2019, pp. 121–132.

[13] A. Karaagac, I. Moerman, and J. Hoebeke, "Hybrid schedule management in 6TiSCH networks: The coexistence of determinism and flexibility," *IEEE Access*, vol. 6, pp. 33941–33952, 2018.

[14] S. Jeong, J. Paek, H.-S. Kim, and S. Bahk, "TESLA: Traffic-aware elastic slotframe adjustment in TSCH networks," *IEEE Access*, vol. 7, pp. 130468–130483, 2019.

[15] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-demand TSCH scheduling with traffic-awareness," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 69–78.

[16] A. Elsts, S. Kim, H.-S. Kim, and C. Kim, "An empirical survey of autonomous scheduling methods for TSCH," *IEEE Access*, vol. 8, pp. 67147–67165, 2020.

[17] G. Carignani, F. Righetti, C. Vallati, M. Tiloca, and G. Anastasi, "Evaluation of feasibility and impact of attacks against the 6top protocol in 6TiSCH networks," in *Proc. Int. Symp. World Wireless, Mobile Multimedia Networks*, Aug. 2020, pp. 68–77.

[18] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "Analysis of distributed and autonomous scheduling functions for 6TiSCH networks," *IEEE Access*, vol. 8, pp. 158243–158262, 2020.

[19] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top) Protocol (6P)*, document RFC 8480, Nov. 2018.

[20] O. Gaddour and A. Koubáa, "RPL in a nutshell: A survey," *Comput. Netw.*, vol. 56, no. 14, pp. 3163–3178, Sep. 2012.

[21] R. Soua, P. Minet, and E. Livolant, "MODESA: An optimized multi-channel slot assignment for raw data convergecast in wireless sensor networks," in *Proc. IEEE 31st Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2012, pp. 91–100.

[22] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3655–3666, Oct. 2013.

[23] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "An experimental evaluation of the 6top protocol for industrial IoT applications," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2019, pp. 1–6.

[24] X. Vilajosana, K. Pister, and T. Watteyne, *Minimal IPv6 Over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*, document RFC 8180, May 2017.

[25] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 641–648.

[26] T. Watteyne, J. Weiss, L. Doherty, and J. Simon, "Industrial IEEE802.15.4e networks: Performance and trade-offs," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 604–609.

**FRANCESCA RIGHETTI** (Member, IEEE) received the master's and Ph.D. degrees in computer engineering from the University of Pisa, Pisa, Italy, in 2017 and 2021, respectively. She is currently a Postdoctoral Research Fellow with the Information Engineering Department, University of Pisa. She took part in national and international projects, including the SmarT INtelliGent RAilwaY (STINGRAY) project with the ISTI-CNR, Pisa, and the "ECOAP: Experimental assessment of congestion control strategies for the Constrained Application Protocol" project. Her research interests include wireless sensor networks, and their applications, the Internet of Things (IoT), and the Industrial Internet of Things (IIoT). She has served as a TPC for different international conference and workshops, including IEEE SMARTCOMP 2020 and 4$^{th}$ International Workshop on Pervasive Smart Living Spaces, PerLS.

**CARLO VALLATI** (Member, IEEE) is currently an Assistant Professor (Tenured) of computer systems engineering with the University of Pisa, Italy. He is the Coordinator of the Cloud Computing, Big Data and Cybersecurity Crosslab, founded in the framework of the Departments of Excellence funded by the Italian Ministry of Education, University and Research. He is the coauthor of more than 50 international publications. His research activities include different areas, such as wireless and sensor networks, protocols and platforms for the Internet of Things, algorithms and architectures for edge/fog computing. He has been involved in different national and international projects, including in particular the FP7-ICT project "BETaaS: Building the Environment for the Things as a Service." He has been the Principal Investigator of the project "ECOAP: Experimental assessment of congestion control strategies for the Constrained Application Protocol," funded with a grant of 45000 euro by the European Project WiSHFUL under the Fifth Open Call for experiments (WiSHFUL-OC5).

**ARIANNA GAVIOLI** received the master's degree *(cum laude)* in embedded computing systems from the University of Pisa, Italy, in June 2021. She prepared her master's thesis under the supervision of Prof. Giuseppe Anastasi. Her main research interests include the Industrial Internet of Things (IIoT), cyber-physical systems, and their applications in the industrial field.

**GIUSEPPE ANASTASI** (Member, IEEE) worked as the (Founding) Director of the CINI (Smart Cities National Laboratory), a nation-wide competence center on smart cities and communities, consisting of 29 nodes (local labs) located at different Italian universities, from 2015 to 2018. From 2016 to 2020, he was the Head of the Department of Information Engineering (DII), University of Pisa. He is currently a Professor of computer engineering with the Department of Information Engineering (DII), University of Pisa, Italy. He is also the Director of the Industry 4.0 CrossLab, funded by the Italian Ministry of Education and Research (MIUR) in the framework of the "Departments of Excellence" Program, which consists of six interdisciplinary and integrated research laboratories (CrossLabs) covering all the key areas of Industry 4.0. He has co-founded many successful international workshops and conferences. He has published more than 150 research articles in the area of computer networking and distributed systems. His publications have received more than 10,000 citations, according to Google Scholar (H-index=43). His current research interests include the Internet of Things, fog/edge computing, cyber-physical systems, cybersecurity, and smart environments. He is also serving as a Steering Committee Member for the IEEE SMARTCOMP Conference. He served as the General Chair for IEEE SMARTCOMP 2018 and IEEE WoWMoM 2005; a Program Chair for IEEE SMARTCOMP 2016, IEEE MSN 2015, IFIP/IEEE SustainIT 2012, IEEE PerCom 2010, and IEEE WoWMoM 2008; and a Vice Program Chair for IEEE MASS 2007. He served as an Area Editor for *Pervasive and Mobile Computing* (PMC, 2007–2016); an Associate Editor for *Sustainable Computing* (SUSCOM, 2010–2015); and an Area Editor for *Computer Communications* (ComCom, 2008–2010). He is a Co-Editor of two books: *Advanced Lectures in Networking* (LNCS 2497, Springer, 2002), and *Methodologies and Technologies for Networked Enterprises* (LNCS 7200, Springer, 2012).