LASA-R: Location-Aware Scheduling Algorithm With Rescheduling for Industrial IoT Networks With Mobile Nodes

Marco Pettorali, *Member, IEEE*, Francesca Righetti, Carlo Vallati, Sajal K. Das, *Fellow, IEEE*, and Giuseppe Anastasi

***Abstract*—The synchronized single-hop multiple gateway (SHMG) framework has been recently proposed to support mobility in 6TiSCH, the network architecture defined by the IETF for the Industrial Internet of Things (IIoT). SHMG includes a scheduling policy to allocate communication resources to mobile nodes (MNs) in order to satisfy the stringent requirements of industrial applications. Current scheduling algorithms, however, manage mobility by simply over-allocating communication resources, without taking into account the position of MNs. In this article, we propose a location-aware scheduling algorithm with rescheduling (LASA-R) that leverages the position of MNs, reported via periodic position notification (PN) messages, to optimize the allocation of communication resources. LASA-R also includes a conflict resolution mechanism to modify the schedule, as conflicts are detected. Finally, a mathematical methodology is developed to determine the optimal PN period. LASA-R is assessed through simulations. The results obtained show that it can guarantee a high reliability and a bounded latency, even with a very large number of MNs.**

***Index Terms*—6TiSCH, conflict avoidance, Industrial Internet of Things (IIoT), mobility, scheduling.**

I. Introduction

TO FOSTER the adoption of the Industrial Internet of Things (IIoT) paradigm, the Internet engineering task force (IETF) has standardized the 6TiSCH architecture [1], an IPv6-based standard architecture that allows to interconnect industrial devices to the Internet, via one or more border routers (BRs), with an industrial grade of service. 6TiSCH leverages the time slotted channel hopping (TSCH) mode of the IEEE 802.15.4 standard for short-range wireless communication [2]. TSCH ensures *time-bounded and predictable latency* via slotted access, *increased network capacity* through multichannel communication, and *improved reliability* thanks to channel hopping. One of the key component of the architecture is the scheduling function (SF) used to allocate communication resources to nodes.

Potential interconnected objects are not limited to stationary devices. Actually, many industrial applications involve mobile objects, such as autonomous vehicles, mobile robots, wearable devices carried by workers, etc., and this trend is expect to grow more and more in the coming years.

However, the 6TiSCH architecture does not include any mechanism for the efficient management of node mobility, and implicitly assumes that nodes are stationary.

In the literature, some previous works [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] have considered node mobility in the IIoT. However, only a few of them deals with the 6TiSCH architecture [6], [7], [8], [9], [10], [11], [12], [13], [14], and none of them considers the definition of a whole framework to handle mobility.

Instead, the synchronized single-hop multiple gateway (SHMG) framework, originally proposed in [4] and then extended in [15], is a promising approach to handle node mobility in 6TiSCH networks. SHMG relies on a centralized approach. With reference to Fig. 1, the network coordinator (NC) is responsible for calculating a communication schedule and allocating communication resources to each mobile node (MN) and BR, based on the application requirements and network conditions. Then, the communication schedule is diffused to all the BRs and MNs in the network.

While many scheduling algorithms can be used within the SHMG framework, in literature only a few are specifically tailored to support mobility. Such solutions, however, adopt a conservative approach where resources are over-provisioned in a static fashion to handle mobility. For instance, the scheduling algorithms proposed in [4] and [15] allocate timeslots in a static and dedicated way in all the BRs, to guarantee the stringent requirements of industrial applications and avoid service interruptions due to mobility. Specifically, the shared-downstream, dedicated-upstream (SD-DU) scheduling [15] allocates dedicated timeslots for uplink

Manuscript received 10 July 2023; revised 7 December 2023; accepted 28 December 2023. Date of publication 12 January 2024; date of current version 25 April 2024. This work was supported in part by the Italian Ministry of Education and Research (MUR) in the framework of 1) the CrossLab and FoReLab Projects ("Departments of Excellence" Program); 2) the PNRR National Centre for HPC, Big Data, and Quantum Computing (Spoke 1, CUP: I53C22000690001); 3) the Project "JOULE: Joint Resource Management in Reconfigurable I4.0 Factories" under Grant 2022TMT4WA; and 4) the Project "CAVIA: Enabling the Cloud-to-Autonomous-Vehicles Continuum for Future Industrial Applications" under Grant 2022JAFATE, and in part by the U.S. NSF under Grant EPCN-2319995, Grant CSSI-2104078, Grant CNS-2008878, Grant SaTC-2030624, and Grant SCC-1952045. *(Corresponding author: Marco Pettorali.)*

Marco Pettorali, Francesca Righetti, Carlo Vallati, and Giuseppe Anastasi are with the Department of Information Engineering, University of Pisa, 56122 Pisa, Italy (e-mail: marco.pettorali@phd.unipi.it; francesca.righetti@unipi.it; carlo.vallati@unipi.it; giuseppe.anastasi@unipi.it).

Sajal K. Das is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 USA (e-mail: sdas@mst.edu).

Digital Object Identifier 10.1109/JIOT.2024.3353532

communication (i.e., from MN to BR), while timeslots for downlink communication are shared by a certain number of MNs. Hence, conflicts can occur in the communication from BR to MN.

To allow an efficient utilization of communication resources, while ensuring the highest possible communication reliability to meet the stringent requirements of industrial applications, in this article we propose a location-aware scheduling algorithm with rescheduling (LASA-R). We assume that MNs are capable of tracking their position and direction, and send them to the NC via periodic position notification (PN) messages. Based on this information, the NC calculates an initial schedule by solving an optimization problem to derive a communication schedule with minimum number of conflicts (i.e., number of MNs that are in the communication range of the same BR with overlapping resource allocated). In order to resolve possible conflicts, either in the initial schedule or arising at run time due to mobility, a conflict resolution mechanism is also included. This mechanism aims at rescheduling the allocated communication resources, as conflicts are detected. In order to optimize the configuration of LASA-R, we also present a mathematical methodology to determine the optimal PN period, in order to ensure the proper balance between accuracy and resource consumption.

The work presented in this article extends a conference work by the same authors [14], where the location-aware scheduling algorithm (LASA) was proposed. Specifically, LASA-R extends LASA by introducing a rescheduling mechanism to manage and resolve conflicts. In addition, its configuration is driven by an analytical model to properly select the period of PN transmissions.

LASA-R is assessed through simulation against LASA [14], A3 [16], and TESLA [17] algorithms. We also considered an optimal (but unfeasible) scheduling algorithm, where an oracle derives the communication schedule based on exact position information. Our results show that LASA-R ensures a very high reliability, close to the one provided by the oracle. The comparison with LASA, in particular, highlights how the conflict resolution mechanism is effective in improving the performance.

The remainder of this article is organized as follows. Section II introduces the 6TiSCH architecture and the SHMG framework. In Section III, we overview the related work. In Section IV, we describe LASA-R. In Section V, we present the analytical methodology for selecting the optimal PN period. In Section VI, we present the simulation setup, while in Section VII we present the results obtained. Finally, in Section VIII, we draw our conclusions.

II. 6TiSCH Architecture

In this section, we introduce the main features of the 6TiSCH architecture [1], as well as the SHMG framework to manage mobility in 6TiSCH networks.

As shown in Fig. 1, a 6TiSCH network is composed by MNs connected to the Internet through BRs. The communication between MNs and BRs is based on the IEEE 802.15.4 TSCH protocol [2]. TSCH provides *time-slotted access* by dividing

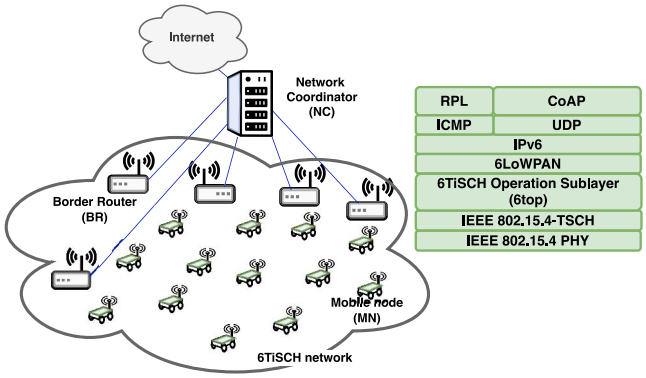

Fig. 1. 6TiSCH reference network and protocol stack.

time in fixed-duration *timeslots*, grouped in a *slotframe* that repeats periodically over time. With the aim of increasing the network capacity, TSCH allows nodes to exploit *multichannel communication*, i.e., different nodes can transmit in the same timeslot exploiting the 16 available channels, each identified by a *channel offset* (an integer value in the range 0–15). Finally, the *frequency hopping* mechanism, that allows nodes to change their operating frequency at each timeslot following a predefined hopping sequence, mitigates the negative effects of multipath fading and interference. In the TSCH slotframe, each *cell* identifies a communication resource through the couple (*timeslot*, *channel offset*). There are two kinds of cells, namely, *dedicated* or *shared*. In dedicated cells the communication is guaranteed to be contention-free among couple of nodes. Instead, shared cells are accessed on a contention basis and, hence, collisions may occurs.

The TSCH standard [2] does not define approaches to allocate cells to nodes for communication. Hence, as shown in Fig. 1, the 6TiSCH architecture has introduced the Operation (6top) sublayer, which is responsible for the allocation of cells to nodes. It relays on an SF and on the 6top protocol (6P) [18]. The SF computes the number of cells each node requires to satisfy its traffic requirements, 6P is then exploited to negotiate the allocation/deallocation of cells among neighbor nodes. On top of the 6top sublayer, the 6LoWPAN adaptation protocol compresses IPv6 datagrams to fit into TSCH frames, the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [19] manages multihop data delivery, and finally, the user datagram protocol (UDP) manages data messages generated by the application.

A. Synchronized Single-Hop Multiple Gateway Framework

The SHMG framework follows a centralized scheme to manage mobility in 6TiSCH networks, and the entities involved are shown in Fig. 1. The NC is the central entity that computes the communication schedule, while BRs act as gateways between the 6TiSCH network and the Internet to provide connectivity to MNs. MNs generate data packets that are transmitted to the closest BR through single-hop communication.

It is noteworthy that, in the context of the SHMG architecture, achieving a high level of reliability for the NC is crucial

and requires it to be consistently powered and operational. However, the implementation details are beyond the scope of this article.

In this framework, each BR builds a star topology with the MNs located in its transmission range. Depending on the scheduling algorithm adopted, each BR installs the communication schedule computed by the NC. The communication schedule may be either synchronized among all the BRs, or it can change based on the actual MNs in the transmission range of each single BR. However, in both cases, in order to handle node mobility efficiently, MNs do not need to acquire cells when moving from one BR to another.

III. Related Work

In the literature, only a few works consider node mobility in 6TiSCH networks. Specifically, the works in [9] and [10] address mobility at the MAC layer, those in [6], [8], and [14] focus on mobility at the 6top layer, while papers in [7], [11], [12], and [13] operate at the routing layer.

The works in [6], [9], and [10] tackle the problem of the presence of MNs in 6TiSCH by mitigating the impact of handovers between BRs, and define solutions to reduce the time taken by MNs to join the network.

In particular, the work in [6] defines a distributed traffic-aware scheduling function (DT-SF) that implements an analytical estimation of the mobility of the neighbors performed by each MN. DT-SF does not require that MNs have additional hardware for determining their position, instead relies on the knowledge of the number of times each MN changed its parent node and the number of packets in their local queue exchanged in DIO packets, i.e., the routing control packets broadcast by RPL. However, DT-SF is not designed to guarantee a target network performance, such as maximum packet loss and de-synchronization time. Hence, the proposed solution may not be suitable for industrial applications. Moreover, the authors only consider networks with a small number of MNs (10% of the total number of nodes).

Nielsen et al. [9] adopted a different approach, specifically focusing on reducing the rejoining time when an MN loses synchronization with the network. They introduce a mechanism for MNs to predict the cell in which the next synchronization packet will be transmitted. This is achieved by maintaining a local state of the last schedule used and setting up a timer to listen for an EB at the expected time if the last used schedule was still accurate. However, the authors do not provide any assurance regarding the QoS achieved by the network.

Finally, Al-Nidawi and Kemp [10] proposed an enhanced version of the TSCH protocol, namely, mobility-aware TSCH (MTSCH). The proposed solution exploits a single frequency for sending enhanced beacons (EBs) to advertise the presence of the network, hence reducing the scanning time performed by MNs to retrieve the network information. In addition, it exploits ACK messages to advertise time synchronization information. As above, the authors do not provide any guarantee on the service achieved by MNs.

The fundamental concept behind the mobile scheduling updated algorithm for time-slotted channel hopping (MSU-TSCH), as outlined in [8], is to assign dedicated cells for each link between any MN and any fixed nodes. These fixed nodes are presumed to be uniformly distributed across the deployment area. Periodically, the fixed nodes broadcast their positions in dedicated cells, enabling MNs to choose the optimal cell for transmitting their packets and minimizing the transmission attempts by the MNs. However, the size of the network considered in the experiments is limited to ten nodes, with only 2 of them being mobile.

The works in [7], [11], [12], and [13] use a different approach, and define a new objective function (OF)—the specific function used by RPL to regulate data forwarding—to handle node mobility. In particular, Kim and Chung [7] proposed a framework to reduce collisions in shared cells for the transmission of control packets, hence minimizing the time needed by the MN to associate with a parent node. Upon a disconnection from the parent node, the MN advertises this to its neighbors, which allocate dedicated cells for transmitting synchronization information and allow the MN to quickly rejoin the network.

Instead, Kim et al. [11] defined the *MobiRPL* framework, and design an OF that exploits information about the received signal strength indicator (RSSI) collected by MNs about their neighbors. Moreover, they define a *mobility detection* model that allows MNs to estimate their speed without the need of external sensors by exploiting the rate of RPL parent changes.

A similar approach is used in [12], where the *MSE-RPL* framework is defined. The proposed solution adopts a dynamic trickle timer to adjust the sending rate of the DIO messages by an MN, which is increased when the MN detects neighbors with high mobility. MNs track the RSSI of the DIO messages received by each neighbor, and rank them according to their estimated relative speed. The neighbor with the lowest variation in the RSSI is selected as parent.

These works, however, only consider small networks with a low number of MNs. Instead, the work in [13] defines the reliable and mobility-aware RPL (ARMOR) framework, which introduces the time to reside (TTR) metric, that provides an estimation of how long the MNs will be in the transmission range of each other. ARMOR requires each MN to obtain its location information (current position and direction), using a positioning system (e.g., GPS or motion sensors), which is then embedded in DIO messages. MNs compute the TTR metric, exploiting the location information periodically obtained by each neighbor, and select the one with the highest TTR as parent. Although the work considers a larger number of MNs, with respect to the previous works, the network performance shown is not suitable for industrial applications, since the packet loss is higher than 40% with a number of MNs equal to 40.

Finally, the work in [14] defines the *LASA*, specifically tailored to the SHMG framework, that exploits the location information of MNs to manage the allocation of communication resources. LASA leverages the same approach used in LASA-R to allocate communication resources in order to guarantee the high reliability required by industrial applications.

However, LASA implements a very simple approach for conflict management, where a single MN is considered for reception by the BR when several MNs happens to be scheduled for transmission in the same timeslot. In addition, LASA does not include any mechanism to resolve conflicts.

LASA-R extends the LASA algorithm by introducing a rescheduling mechanism that resolves conflicts as they occur at run time, improving the performance in a very significant way. In addition, in LASA-R the transmission period of PN can be optimized, depending on the specific scenario, using an analytical model, which allows a tradeoff between overhead and schedule accuracy.

IV. LASA-R Description

In this section, we describe the LASA-R that is executed on the NC. After the network bootstrap, the NC computes the initial schedule, based on the location of MNs within the deployment area. The computed schedule is then installed on the MNs and BRs.

With LASA-R, cells are activated/deactivated on BRs, at run time, depending on the mobility of MNs, in such a way to guarantee consistent connectivity. LASA-R also includes a rescheduling mechanism that modifies the schedule whenever a conflict is detected. A conflict arises every time two or more MNs, that are in the communication range of the same BR, are scheduled for transmission in the same timeslot (although at a different channel offset), since we assume that a BR can listen to only one channel at time. The removal of conflicts is paramount to improve the reliability, as conflicts result in packet loss.

In the following, we provide a detailed description of the proposed algorithm. Specifically, Section IV-A describes how the schedule is computed initially. Section IV-B shows how cells are activated on BRs, depending on the mobility of MNs. Section IV-C introduces the general approach to conflict management, while Sections IV-D and IV-E describe the two main components of the conflict management mechanism, namely, the selection algorithm, to select the conflicts to be resolved at each slotframe (Section IV-D), and the rescheduling algorithm that modifies the current schedule to remove conflicts (Section IV-E).

A. Initial Schedule Computation

LASA-R computes the initial communication schedule just after the network bootstrap. To this aim, it is assumed that the NC knows the traffic requirements of all the MNs and their initial location. Traffic and location information are collected by the NC during a preliminary phase. How this process is performed is considered out of scope in this article. It is also assumed that the packet generation rate, r (expressed in packet/sec), is the same for all the MNs.

First, LASA-R computes the slotframe length S (in number of timeslots) as the maximum length that can satisfy the packet generation rate of MNs. Specifically, LASA-R allocates 1 cell every $1/r$ s to each MN. Thus, S is computed as $S = \lfloor 1/(T_s \cdot r) \rfloor$, where T_s is the timeslot duration in seconds (for instance, a common value of T_s is 0.015 s).

Subsequently, knowing the slotframe length S, and the initial location of all the MNs, LASA-R computes the optimal communication schedule. To this end, in [14] we defined an optimization problem, namely, the minimum-conflict schedule (MCS) problem, to compute the optimal communication schedule, i.e., the schedule that minimizes the number of conflicts. Hence, the NC derives the initial schedule by solving the MCS problem.

This schedule is then installed on all the MNs and, hence, all the MNs have the same schedule. The schedule is also installed on the BRs. However, at each BR, *only* cells allocated to the MNs that are located within its communication range will be marked as active, while the remaining ones will be marked as inactive. The BR will listen to receive possible packets from MNs *only* on active cells.

B. Cell Activation for Mobility Management

During the operational phase, node mobility is managed by LASA-R to guarantee connectivity and continuity of service, also when MNs change their position and BR. The complexity of the mobility management process is handled by the NC, which is responsible for activating preallocated cells on the schedule installed on the BRs.

To this aim, the NC exploits the location information (i.e., position and direction) that is periodically embedded in data packets sent by MNs, exploiting the *PN* field that includes an approximation of the position and direction of the MN, as described in [14]. Moreover in [14], we have demonstrated that this information can be compressed to 2 bytes without compromising performance. We denote by P_{PN} the transmission period of PN, expressed in terms of number of packets. For instance, if $P_{\mathrm{PN}} = 5$, the location information is sent every five data packets. Using the position and direction of an MN, the NC derives its trajectory and activates (preallocated) cells on the BRs that are located along the path of that MN. This ensures, in a preventive manner, the seamless connectivity of the MN. Similarly, cells are deactivated when the MN moves away from a certain BR.

Finally, in order to improve the communication reliability, we introduce the backup allocation strategy (BUAS). It enables the utilization of timeslots with temporarily no active cells, such as when no MN is within the communication range of the BR. BUAS is executed on each BR at the beginning of each timeslot. If no active cells are available, the BR requests the identifier of the nearest MN from the NC with a cell allocated in that timeslot. Subsequently, the BR activates the cell for the MN, enabling the reception of packets from that MN.

C. Conflict Management

The initial schedule, based on the position of MNs at network bootstrap, may result in conflicts as the MNs start moving. This could happen, for instance, when a node A moves to a BR that is serving another node B using a cell corresponding to the same timeslot as node A. Since the BR cannot receive on two different frequencies, during the same timeslot, a conflict occurs. Moreover, conflicts might also occur on the initial schedule, as the algorithm presented in

Section IV-A aims at minimizing conflicts, however, it is not conflict-free.

Upon detecting a conflict on a BR, the NC instructs the BR to select a single MN (i.e., cell), among the conflicting ones, for data reception, so that at least one transmission can be correctly received. In our implementation the NC adopts the *Closest-First* policy, i.e., it selects the MN that is located closet to the BR. It has been shown in [14] that this policy performs significantly better than other alternative ones, such as FIFO or Random.

On the other hand, to resolve a conflict, the NC needs to modify the schedule. Changes in the schedule, after the initial schedule, are notified to the MNs via a dedicated control message, namely, schedule update (SU). Specifically, SU messages are generated by the NC and transmitted to the BRs, and contain the list of (MN ID, new cell) pairs of the subset of MNs that must update their cell allocation. Then, BRs broadcast them to MNs using a dedicated timeslot and channel offset. While the timeslot is the same for all the BRs, different channel offsets are used, by different BRs, in order to avoid collisions with neighboring BRs. Moreover, BRs situated at a distance from each other are allowed to reuse the same channel offset. The channel offset for SU transmission used by each BR is assumed to be known by the MNs since network bootstrap, and it remains fixed during the operational phase.

Since SU messages can get lost or corrupted, a schedule change issued by the NC may not be received correctly by one or more MNs, thus resulting in a schedule inconsistency. In order to detect inconsistencies, the NC monitors the data transmissions from MNs. If an MN does not transmit any data packet for a consecutive number of times, it is assumed that the MN has lost a previous SU and, hence, its schedule is not aligned with the last version. In order to resynchronize the schedule, the NC retransmits the SU message to the MN. Throughout, this operation will be referred as a *refresh* transmission.

The amount of information that can be transmitted on a single SU message is limited, due to size of TSCH frames. Consequently, it may not be possible to include all the schedule changes in a single SU message. Hence, a limit is set to the number of MNs that can be rescheduled, and notified through a single SU, for each slotframe. To this aim, we define a selection algorithm (SA) to select the MNs to reschedule at the beginning of each slotframe. Then, the *Rescheduling Algorithm* is invoked to resolve conflicts and compute the new schedule. Finally, interested MNs are notified through the SU message.

In the remainder of this section we first present the *SA* in Section IV-D and, then, the Rescheduling Algorithm in Section IV-E.

D. Selection Algorithm

The SA is executed by the NC at the beginning of every slotframe to select the (subset of) MNs to reschedule or refresh. Specifically, the NC detects the MNs that are involved in a conflict by comparing information on MN positions with the current schedule. Instead, it detects that an MN that is no longer synchronized with the current schedule by monitoring its transmissions over time. If no successful transmission is performed by the MN for a number of T_{WAIT} consecutive slotframes, the MN is assumed to be de-synchronized and marked for a refresh.

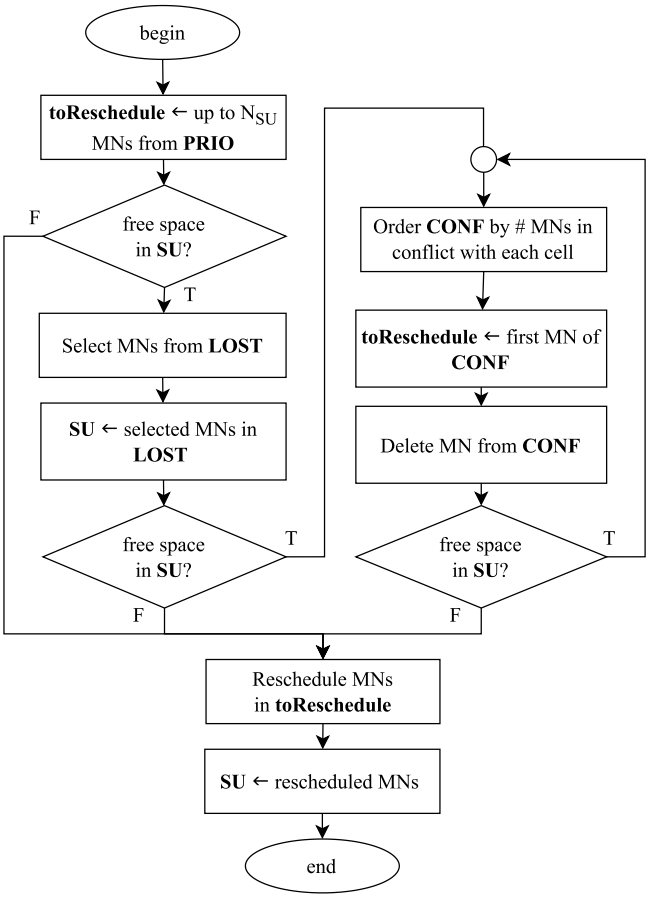

Fig. 2. Flowchart of the selection algorithm.

SA takes, as input, the following information: 1) current schedule, *currSchedule*; 2) maximum number of updates that can be included in an SU, N_{SU}; 3) the slotframe length, S; 4) coordinates of BRs, D; and 5) status of each MN (e.g., position, allocated cell, and last packet successfully sent), $INFO_{MNs}$. The result produced by SA is the list of updates to be included in the next SU message *updatesNextSU*.

SA considers three sets of MNs.

1) The *PRIO* set contains MNs that are both in conflict and not transmitting for more than T_{WAIT} slotframes—these MNs should be considered for both a reschedule and a refresh.
2) the *LOST* set includes MNs that have lost their synchronization with the current schedule and need a refresh.
3) The *CONF* set includes the remaining MNs in conflict.

The overall workflow is depicted in Fig. 2. The algorithm selects cells (i.e., MNs) with a strict priority among the different sets. Specifically, it first considers the *PRIO* set, as it includes MNs that are involved both in a conflict and in a de-synchronization. Then, it considers the *LOST* set, i.e., MNs marked for refresh. Finally, it considers the *CONF*

set, i.e., MNs involved in a conflict. The algorithm selects MNs from the next set only if there is remaining space in the SU. In case the remaining space in the SU is not sufficient, the MNs of a set are selected on a given order, namely, a random order for *PRIO* and *LOST*, and on a priority-based order for *CONF*. For the sake of fairness, MNs in *CONF* are prioritized based on the current number of conflicts in the same timeslot. Specifically, the timeslot with the highest number of MNs in conflict is selected first, and a single MN is randomly selected. Then, the process is repeated, and the list of conflicts is updated accordingly, until either the SU is full or there are no more MNs in *CONF*.

Once all the MNs have been selected, the Rescheduling Algorithm is invoked to compute the new allocation, in order to minimize the number of conflicts in the new schedule.

E. Rescheduling Algorithm

The rescheduling algorithm takes a maximum distance (MD) approach to reschedule the selected MNs. Specifically, if the number of MNs in the network, M, is lower than the number of timeslots available for data transmission in the slotframe, each MN has a dedicated timeslot and conflicts never occur. Under the assumption that M is larger than the slotframe length, for each timeslot, there will be two or more cells allocated at different channel offsets. Hence, reallocating a cell to resolve a conflict creates a new potential conflict. The basic idea of the MD approach is to select the new cell in such a way to minimize the probability that the conflict will actually occur in the future. To this end, the new cell is allocated in the timeslot of the MN whose physical distance, from the MN to reschedule, is maximum.

The MD algorithm is detailed in Algorithm 1. It receives the following input parameters: 1) number of MNs, M; 2) slotframe length, S; 3) current cell allocation, *currSchedule*; 4) list of cells to reschedule, *cellsToReschedule*; and 5) position of each MN POS_{MNs}. The algorithm computes the new timeslot and channel offset for all the MNs to reschedule in *rescheduledCells*.

Initially, the algorithm initializes the *posTs* vector with the position of the MNs that are not involved in a conflict (from line 1 to 7). This vector is then used to compute the physical distance between the MN to reschedule and all the other MNs without conflicts, in order to select the cell of the MN at the MD.

After initializing the *posTs* vector, the algorithm iterates over the cells to reallocate (lines 9–23). For each cell, the position of corresponding MN is derived (line 11). Then, for each timeslot s in the slotframe, the position of the closest MN p_c with a cell allocated in s is computed (line 14), and the distance $dist_s$ between the MN to reschedule and p_c is calculated (line 15). At each iteration, the algorithm reschedules MN in a new cell; specifically, the new cell is allocated in the timeslot corresponding to the MD $dist_s$ (line 17). After rescheduling, the position of the node is added to *posTs*, so that its position will be taken into account in the next iterations.

Algorithm 1: MD Rescheduling Algorithm

Input:
M = number of MNs
S = slotframe length
currSchedule = current list of allocated cells
cellsToReschedule = cells to be rescheduled
POS_{MNs} = list of positions of the MNs

Output:
rescheduledCells = list of the cells to be inserted in an SU

```
1  posTs ← {posTs_0:[ ], ..., posTs_{S−1}:[ ]}
2  for c in currSchedule not in cellsToReschedule do
3     Find MN that has cell c allocated
4     p_MN ← position of the MN from POS_MNs
5     ts ← timeslot of cell c
6     Insert p_MN in posTs_ts list
7  end
8
9  for c in cellsToReschedule then
10    Find MN that has cell c allocated
11    p_MN ← position of the MN from POS_MNs
12    dist ← {dist_0 : ∞, ..., dist_{S−1} : ∞}
13    for s = 0 to S − 1 do
14       p_c ← closest element of posTs_s to p_MN
15       dist_s ← ‖p_c − p_MN‖
16    end
17    ts ← element of dist with maximum value
18    Insert p_MN in posTs_ts list
19    if ts ≠ timeslot of c then
20       c_res ← new cell allocated to MN in timeslot ts
21       Insert c_res in rescheduledCells
22    end
23 end
```

It is worth to mention that the timeslot selected by the MD algorithm may not be different from the current one. In this case, there is no rescheduling and the algorithm proceeds with the next cell to reallocate. Otherwise, the algorithm reallocates the cell in the selected timeslot, at a random channel offset (lines 19–22).

In order to assess the performance of MD we run a set of simulations to compare it against the MCS algorithm used to derive the initial schedule (see Section IV-A). Since the MCS produces an optimal solution that minimizes conflicts between MNs, the comparison can help to measure how the heuristic solution produced by MD is far from the optimal solution. In particular, we are interested in comparing the fraction of conflicts in the schedules generated by the two algorithms and their average computation time.

It is worth to highlight that MD has been conceived to reschedule conflicting MNs, typically, a subset of all the MNs in the network, while MCS is used in Section IV-A to produce an initial schedule and, hence, it considers all the MNs. In

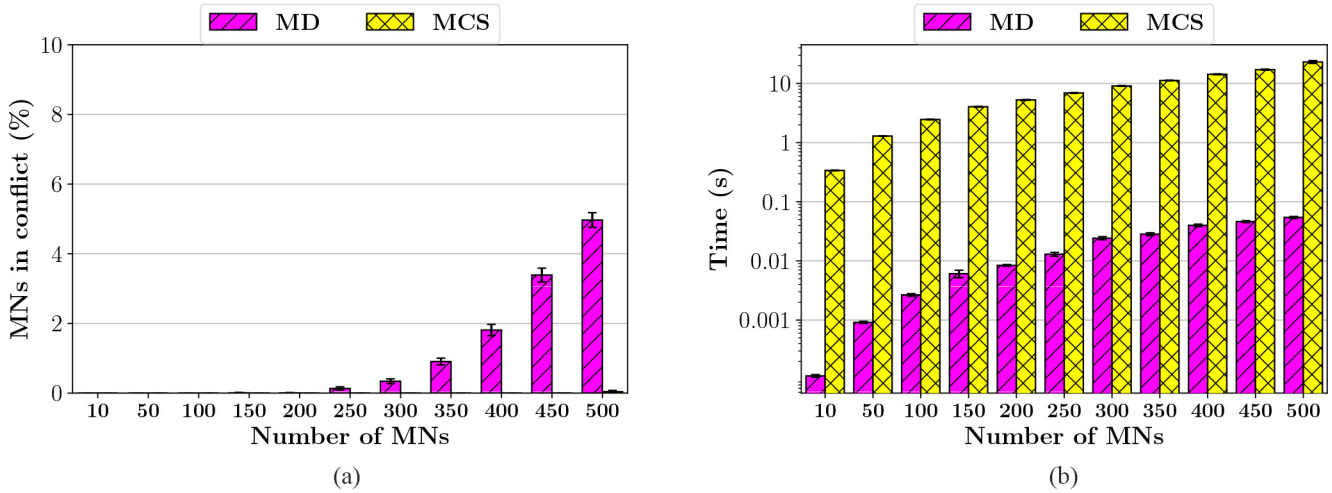

Fig. 3. Comparison between MD and MCS in terms of fraction of MNs in conflict after the execution of the algorithm and average computation time. (a) Fraction of MNs in conflict. (b) Average computation time.

order to make the comparison fair, in our evaluation we assumed that all the MNs are marked for rescheduling and, hence, the two algorithms consider the same number of MNs.

The two algorithms are implemented in an ad-hoc C++ simulator. MCS is implemented using the Google OR-TOOLS library.[1] In our simulations, we consider different scenarios with an increasing number of MNs (from 10 to 500), randomly deployed in a square area of 400 m $\times$400 m. In order to ensure statistically sound results, each scenario was repeated 100 times. For each scenario, an initial cell allocation is generated randomly and, then, both MCS and MD are executed to calculate a global schedule. For both algorithms, we measure the number of conflicts in the generated schedule and the average computation time taken to generate it.

Fig. 3 shows the percentage of conflicts for the two algorithms with different numbers of MNs. We can see that MCS typically generates a conflict-free schedule, even when the number of MNs is very large, while MD generates a suboptimal schedule with a larger fraction of conflicts. However, the number of conflicts generated by MD is negligible up to 200 MNs and, even with 500 MNs, it is around 5%, i.e., a very limited fraction.

Fig. 3 shows the average computation time of the two algorithms. Even when the number of MNs is very large (e.g., 500), the MD algorithm is able to generate a complete schedule in approximately 70 ms. On the other side, the computation time required by MCS is much larger and ranges from 300 ms (with ten MNs) to tens of seconds (with 300 MNs). A so large computation time is not acceptable for on-line rescheduling that occurs very frequently (i.e., at each slotframe). Instead, the very low computation time of the MC-MS algorithm, coupled with the limited number of conflicts, makes it a very good compromise for our purposes.

[1]https://developers.google.com/optimization

TABLE I
LIST OF INPUTS OF THE P_{PN} MODEL

Symbol	Meaning
M	Number of MNs
v	Speed of the MNs
r	Packet generation rate
B	Number of BRs
D	Set of coordinates of the BRs
S	Slotframe length
N_{SU}	Maximum number of cells in a SU

V. ANALYTICAL METHODOLOGY TO SET P_{PN}

In order to implement the mobility and conflict management functionalities included in LASA-R, the NC requires to receive continuous updates on the position of MNs. To this aim, each MN periodically communicates its position through the PN field included in data packets. We denote by P_{PN} the sending interval of the PNs.

The selection of P_{PN} is based on a tradeoff between accuracy and resource consumption. Specifically, a low P_{PN} value guarantees a high accuracy in the position estimation, which can ensure a better performance. On the other hand, sending frequent notifications increases the overhead, in terms of bandwidth and, above all, energy consumption, since the MN needs to collect information about its position more frequently.

In this section, we present an analytical methodology to compute the appropriate P_{PN} value. We assume that MNs move within the deployment area using the *random pathways* mobility model. In this model, MNs move along a line toward a uniformly distributed point in the area. Once they reach that point, they choose another random point and move in a straight line toward it, repeating this process. We selected this mobility pattern as it represents a challenging scenario with unpredictable and nondeterministic movements.

As a preliminary step, we define an analytical model to derive, mathematically, the packet delivery ratio (PDR) achieved by an MN, depending on operating parameters. PDR is defined as the ratio between the number of packets

correctly received at destination and the total number of packets generated by an MN. Table I summarizes the input parameters of our model. The resulting PDR depends on the distance D_{PN} traveled by an MN between two subsequent PN transmissions. In turn, D_{PN} depends on the MN speed, data transmission rate r, and P_{PN}, as follows:

$$D_{\text{PN}}(P_{\text{PN}}) = v \cdot \frac{1}{r} \cdot P_{\text{PN}}. \tag{1}$$

In addition, the PDR also depends on the number of MNs in conflict, as well as on the impact of the BUAS (described in Section IV-B). In order to master the complexity of the overall system, we consider the different aspects of the system separately, and derive an analytical formulation for each of them. Then, we will integrate the different parts into a final closed formula.

Initially, we focus on the *communication success probability* of a single MN, i.e., the probability of an MN to communicate with the associated BR in the period between two consecutive PN transmissions. We assume that the communication success probability only depends from the positions of the MN and the BR in the area of deployment at the time of the transmission, and is known for each point in the area. The communication success probability decreases as the D_{PN} increases, since: 1) the MN may move away from the currently associated BR and 2) the NC cannot trigger the activation of the cell associated with the MN on a different BR until the next PN transmission.

To derive our model, we are interested in the *average communication success probability*, i.e., the communication success probability averaged over the whole deployment area $\mathcal{A}$, which is provided by the following lemma.

Lemma 1: The average communication success probability of a single MN $E_{\mathcal{A}}(D_{\text{PN}})$ is given by

$$E_{\mathcal{A}}(D_{\text{PN}}) = \frac{1}{\|\mathcal{A}\|} \int_{p \in \mathcal{A}} E_c(p, D_{\text{PN}}) \tag{2}$$

where $\|\mathcal{A}\|$ is the area of $\mathcal{A}$, and $E_c(p, D_{\text{PN}})$ is the average communication success probability restricted to the circular area centered in a generic point p with radius D_{PN} when only the BR associated with p is active.

Proof: Fig. 4 shows a graphical proof of the lemma. We denote by p the point in which an MN transmits a PN. Since the mobility pattern of the MN is random, the point p is uniformly distributed in $\mathcal{A}$. Until the next PN transmission, the MN can be anywhere in a radius D_{PN} from p, which is represented in red in Fig. 4. Hence, the average communication probability of an MN with its associated BR (represented in blue) is given by the mean of the function $E_c(p, D_{\text{PN}})$ over $\mathcal{A}$. ∎

Now, we estimate of the number of MNs in conflict. To this end, we assume that MNs and BRs are uniformly distributed within the deployment area, and that cells are allocated to MNs with uniform distribution within the slotframe. We also assume that the rescheduling algorithm removes, at each execution, a number of conflicts equal to N_{SU}, i.e., the maximum number of notifications that can be accommodate in an SU. Under these assumptions, Lemma 2 the average number of MNs in conflict.

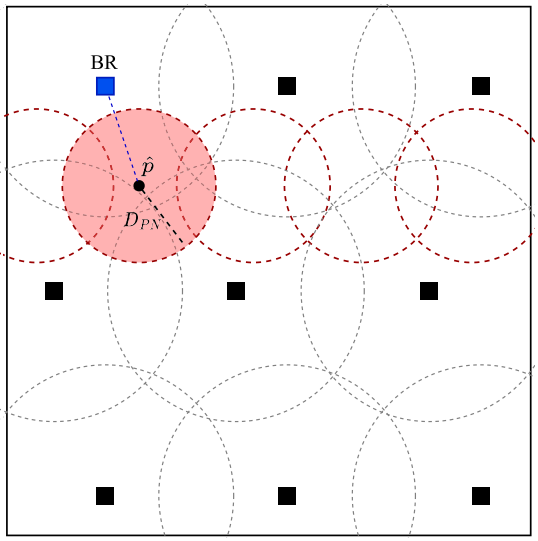

Fig. 4. Impact of D_{PN} on the PDR of an MN.

Lemma 2: The average number of MNs in conflict M_c in the whole network is given by

$$M_c = \max\left(0, B \cdot S \cdot \sum_{x=1}^{M-1} x \cdot p_n(x+1) - N_{\text{SU}}\right). \tag{3}$$

Proof: Initially, we consider a single BR and one timeslot. For each MN, the probability of: 1) being in the range of that BR and 2) having a cell allocated in that timeslot is $1/(B \cdot S)$. Hence, for x MNs, the probability $p_n(x)$ to be in the range of the same BR, with a cell allocated in the same timeslot, is given by the following binomial expression:

$$p_n(x) = \binom{M}{x} \left(\frac{1}{B \cdot S}\right)^x \left(1 - \frac{1}{B \cdot S}\right)^{M-x}. \tag{4}$$

Now, we derive the number of MNs in conflict (or conflicts, for short). Since each BR can serve at most one MNs per timeslot, the number of conflicts, for each timeslot, is equal to the number of MNs that cannot be served by the BR. For instance, if there are $x+1$ MNs in the range of the same BR that are using the same timeslot, there will be x conflicts in that timeslot. Hence, the probability $p_c(x)$ of having x conflicts is $p_c(x) = p_n(x+1)$. Consequently, the average number of conflicts, for a certain timeslot, is the expectation of p_c

$$M_{c,BR} = E\big[p_c(x)\big] = \sum_{x=1}^{M-1} x \cdot p_n(x+1). \tag{5}$$

When considering the whole network, we need to sum up the number of conflicts for each timeslot in the slotframe and for each BR in the deployment area. Hence, the average number of MNs in conflicts in the network is

$$M_{c,NET} = B \cdot S \cdot M_{c,BR} = B \cdot S \cdot \sum_{x=1}^{M-1} x \cdot p_n(x+1). \tag{6}$$

Finally, if we consider the impact of the rescheduling algorithm, which removes N_{SU} conflicts at each execution, the

average number of conflicts in the network can be expressed as follows:

$$M_c = \max\left(0, M_{c,\text{NET}} - N_{\text{SU}}\right). \tag{7}$$

■

Finally, we consider the impact of the BUAS mechanism. We recall that, for each timeslot in the slotframe, a BR can be involved in the BUAS if it has no MN in its range with a cell allocated in that timeslot. To take into account the impact of the BUAS mechanism, we leverage the following two lemmas, and derive: 1) the expected number of BRs, involved in the BUAS, that can receive packets from an MN (Lemma 3) and 2) the average communication probability of the MN with any BR involved in the BUAS (Lemma 4). The assumptions are the same as above.

Lemma 3: The average number of BRs, involved in the BUAS, that may receive packets from an MN, is given by

$$B_b = \frac{B \cdot p_n(0)}{M/S}. \tag{8}$$

Proof: By definition, for any timeslot t, the probability that a BR is involved in the BUAS in timeslot t is equal to the probability that no MN with a cell allocated in t is in its range, i.e., $p_n(0)$, according to Lemma 2. Hence, the average number of BRs involved in the BUAS in timeslot t is then given by $B_{b,t} = B \cdot p_n(0)$.

Since cells allocated to MNs are uniformly distributed in the slotframe, the average number of MNs allocated in the same timeslot is $M_t = M/S$. Finally, as MNs and BRs are uniformly distributed in the deployment area, each MN is listened by $B_{b,t}/M_t$ BRs, from which the result follows. ■

Lemma 4: The average communication probability p_b between a BR involved in the BUAS and an MN is given by

$$p_b = \frac{1}{B}\sum_{i=0}^{B-1} p_{b,i} \tag{9}$$

where $p_{b,i}$ is the average communication probability with the ith BR over all the points of the deployment area.

Proof: Since MNs and BRs are uniformly distributed in the deployment area, and each BR can be involved in the BUAS, the average communication probability p_b between the MN and any BR involved in the BUAS can be computed as the average communication probability $p_{b,i}$ over all the BRs in the network. Hence, the result follows. ■

Based on Lemmas 2–4, we are now in the position to derive the PDR.

Theorem 1: The PDR experienced by an MN is given by

$$\text{PDR}(P_{\text{PN}}) = \text{PDR}_c + \text{PDR}_{nc}(P_{\text{PN}}) \tag{10}$$

where:

1) $\text{PDR}_c = (M_c/M) \cdot [1 - (1 - p_b)^{B_b}]$;
2) $[t]\text{PDR}_{nc}(P_{\text{PN}}) = [1 - (M_c/M)] \cdot [1 - (1 - p_b)^{B_b} \cdot (1 - E_a(D_{\text{PN}}(P_{\text{PN}})))]$.

Proof: We define the fraction of MNs in conflict, with respect to the total number of MNs, as $M_{c,\text{ratio}} = M_c/M$, where M_c is the average number of MNs in conflict in the network, given by Lemma 2.

An MN in conflict can communicate with only the BUAS BRs. Leveraging Lemmas 3 and 4, we can compute the communication success probability of the MN with at least one BUAS BR as $1 - (1 - p_b)^{B_b}$. Hence, the contribution to the overall PDR, due to the MNs in conflict, is given by the expression at point 1) above.

Similarly, we can compute the contribution to the PDR of the MNs that are not in conflict. The fraction of MNs that are not in conflict is $1 - M_{c,\text{ratio}}$. They communicate with the BR in their communication range with a probability depending on the D_{PN}, according to Lemma 2. Moreover, they communicate with the BUAS BRs. Using (1), the contribution of the MNs that are not in conflict is given by the expression at point 2) above. ■

Theorem 1 provides a closed formula of the PDR, as a function of P_{PN}. Hence, it allows us to compute the "optimal" notification period P^*_{PN}, i.e., the largest P_{PN} value that ensures a PDR above a predefined threshold P^*. To this end, the following inequality must hold:

$$\text{PDR}\left(P^*_{\text{PN}}\right) \geq P^*. \tag{11}$$

Consequently, P^*_{PN} can be derived from (11) as follows:

$$P^*_{\text{PN}} = \left\lfloor \text{PDR}^{-1}\left(P^*\right) \right\rfloor \tag{12}$$

where PDR^{-1} can be computed, numerically, using the bisection method.

As a final remark, it is worthwhile highlighting that the target PDR (i.e., P^*) might be unfeasible to reach in some cases, i.e., no value of P_{PN} satisfies (11). For instance, this may happen in a network with a high number of MNs, where the number of conflicts is very large. In such a case, our method selects $P^*_{\text{PN}} = 1$ to guarantee the highest PDR possible.

A. Model Validation

In order to validate the analytical methodology presented above, we ran a set of simulations to compare the P^*_{PN} derived analytically with the optimal P_{PN} value obtained from a set of simulations. To this aim, we exploited the *Mobile-6TiSCH* simulator [20]. More details about the simulator and its settings will be provided in Section VI.

For each considered scenario, we derived the P_{PN} through an exhaustive search, i.e., we ran different simulation experiments with varying P_{PN} values and, then, we selected the largest P_{PN} that results in a PDR, i.e., percentage of packets correctly delivered, above $P^* = 99\%$. The value P^*_{PN}, instead, was derived analytically, according to the formulas in Section V, using a tolerance of 0.5 mm for the bisection method.

We considered different scenarios, characterized by different numbers of MNs (from 10 to 500) and speed (2 and 5 m/s). The transmission rate was set to $r = 2$ pkt/s, resulting in a slotframe length $S = 33$, according the procedure presented in Section IV-A. The number of BRs was $B = 40$, deployed in a square area of 400 m×400 m.

Fig. 5 shows the results of the comparison. We observe that the analytical model is able to estimate the optimal P_{PN} with a good accuracy. When the number of MNs is very large

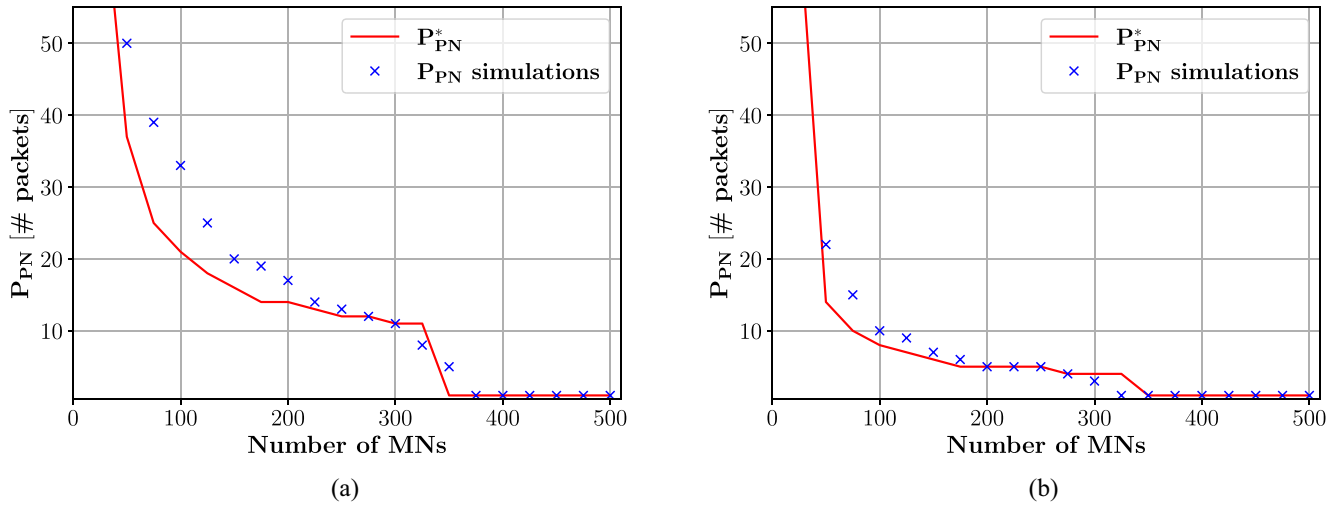

Fig. 5. Comparison between analytical P^*_{PN} and optimal P_{PN} derived by simulation. (a) v = 2 m/s. (b) v = 5 m/s.

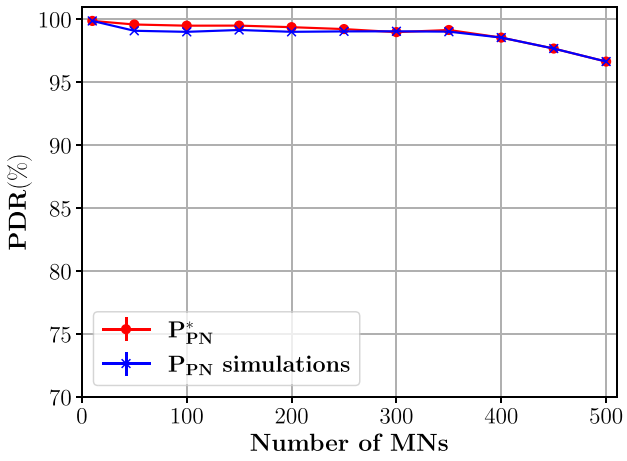

Fig. 6. PDR obtained with analytical P^*_{PN} and optimal P_{PN} derived by simulation ($v = 2$ m/s).

(i.e., 350 or more) both the analytical model and simulation provide the same result, i.e., $P^*_{PN} = 1$. This is because, with such a large number of MNs, the number of conflicts is higher than the maximum number of notifications that can be accommodated in an SU message. Hence, it is necessary to provide the MN position at each data transmission.

The discrepancy between the analytical and simulation results, when the number of MNs is lower than 200, is due to the fact that, for the sake of tractability, the analytical model does not capture all the aspects of the real system, such as the impact of the refresh procedure (see Section IV-C). Also, it is worth mentioning that, the analytical method takes a conservative approach and tends to underestimates the P^*_{PN}, which guarantees a slightly larger PDR, as shown in Fig. 6.

The results presented above confirm the validity of our analytical approach to estimate the optimal P_{PN}. Looking at the trend of the curves in Fig. 5, obtained with $v = 2$ m/s and $v = 5$ m/s, we can observe that the P^*_{PN} value is strongly influenced by the speed of the MNs. When an MN moves at higher a speed (e.g., 5 m/s) its position changes more frequently and hence, position information must be sent with a lower P_{PN}.

VI. Simulation Setup

In this section, we present the simulation setup used for the performance evaluation of LASA-R. For our simulation analysis, we rely on the *Mobile6TiSCH* simulator[2] [20], which is based on OMNeT++.[3] It implements the SHMG architecture and supports various mobility patterns for MNs. Moreover, Mobile6TiSCH implements a realistic channel model to simulate the impact of the wireless channel on the communication between MNs and BRs, including effects, such as multipath fading and interference.

In order to compare the results obtained with LASA-R with existing scheduling algorithms, we have implemented in Mobile6TiSCH the A3 [16] and TESLA [17] algorithms.

In our experiments, we consider a deployment area of 400 m $\times$ 400 m, in which 40 BRs are deployed according to the policy defined in [15]. The number of MNs in the network ranges from 10 to 500. We consider two mobility patterns for the MNs, namely, 1) random waypoints (referred as *random* in the rest of the section), previously presented in Section V and 2) *linear*, in which MNs are programmed to always move along horizontal or vertical lines within the deployment area. We also consider two different speeds for MNs, namely, 2 and 5 m/s. We assume that all the MNs in the network have the same mobility pattern and speed. Regarding the traffic pattern, we assume a constant bitrate traffic from the MNs to the NC, with a packet generation rate r set to 2 pkts/s for each traffic source. The slotframe length S is 33, which is set according to the approach described in Section IV-A based on the packet generation rate. It is worth to highlight that, since one timeslot is reserved for SU transmissions, only 32 timeslots are available for data communication.

In our simulations, we assumed that MNs are able to retrieve their actual positions with exact precision. Nonetheless, we

[2]https://github.com/marcopettorali/Mobile6TiSCH
[3]https://omnetpp.org/

conducted experiments where a certain error was introduced in the position readings.

The maximum number of reschedule/refresh notifications that can be included in an SU N_{SU} is set to 40. This value is selected considering that the total number of cells available for data communication in each 2-D sloframe is equal to $S \cdot N_c$, where N_c is the number of available channels (at most, 16). Hence, every cell can be identified by a 18-bit identifier, where the most significant 9-bits identify the specific cell among the cell in the 2-D slotframe, while the least significant 9-bits identify the MN using that cell for data communication to the NC. Assuming a maximum payload size of 90 bytes, the maximum number of cells that can be accommodated in an SU is equal to $\lfloor (90 \cdot 8/18) \rfloor = 40$.

In our experiments, we measured the PDR, defined as the ratio between the number of packets correctly received by the NC and the total number of packets generated by MNs.

The PDR is a key performance indicator in IIoT environments, as industrial applications typically require high reliability and/or low and predictable latency. In our simulation analysis, we do not consider latency as, in LASA-R, each MN is assigned a dedicated cell for communication with the BR (and conflicts are detected and resolved at run time), while BRs are assumed to be connected with the NC though a high-capacity and reliable wired network (e.g., Ethernet). Hence, the latency experienced by data packets is deterministic, and its upper bound can be easily computed analytically.

To ensure statistically sound results, we run ten independent replicas of each simulation experiment, each of 1000-s duration. The results presented below show the confidence intervals, calculated with a 95% confidence level.

VII. Performance Evaluation

In this section, we present the results of the performance evaluation of LASA-R. Specifically, we analyze the PDR provided by the proposed algorithm in a number of scenarios characterized by different number of MNs, different mobility pattern, and different speed for MNs.

To compare LASA-R with other existing solutions, in our analysis we also consider the LASA algorithm proposed in [14]. This allows to understand the impact of rescheduling and refreshing on the PDR achieved by MNs. Since the performance of both protocols strongly depends on the frequency of PN notifications (i.e., the P_{PN} value), in our evaluation we assume that, for LASA, P_{PN} is set to 1, which ensures the best performance. Instead, in LASA-R P_{PN} is configured according to the procedure presented in Section V.

For comparison, we also considered two autonomous scheduling algorithms, namely, A3 [16] and TESLA [17]. These algorithms are typically used in scenarios with stationary nodes and multihop communication, where the traffic load on each node may vary over time. Hence, we adapted them to the specific scenario considered in this article.

A3 computes the number of cells to be allocated for each node and delegates the allocation to a proper scheduling algorithm. In our setup, we configured A3 to delegate scheduling to ALICE [21], as it is the most suitable option for our scenario. ALICE allocates a dedicated cell, for each upstream link, which position in the slotframe is obtained through a hash function taking, as input parameters, the addresses of the transmitter and receiver, and the absolute slotframe number (ASFN), i.e., the number of slotframes elapsed since the network bootstrap. In particular, BRs are configured with one reception cell for each MN in their communication range, while MNs are assigned with one transmission cell for communication with their closest BR. In our simulations, packet retransmissions are disabled since the considered scenarios involves real-time communication, and old data becomes stale at the next transmission opportunity. The packet generation rate for the MNs is then constant, and equal to 2 pkts/s. Consequently, the *Traffic Load Estimation* mechanism is disabled on both BRs and MNs, as the traffic load is predictable and the *Resource Allocation* algorithm is never triggered. As a result, for each MN-BR pair, only one cell is allocated for upstream communication. Control packets, such as DIO and DAO packets, are assumed to always be transmitted successfully.

Unlike A3, TESLA is built on top of the Orchestra (receiver based) scheduling algorithm [22]. Accordingly, each node allocates one cell for reception at the beginning of the slotframe. Nodes periodically estimate the inward traffic rate and adjust the slotframe size, reducing it when the estimated traffic load is high, and increasing it otherwise. In this work, we configured TESLA to allocate reception cells on BRs only, as the considered traffic scenario is upstream only. As above, MNs allocate one transmission cell for communicating with their closest BR only, and packet retransmissions are disabled. Finally, we assume that control packets are always transmitted successfully and consider the parameter values proposed in the original paper [17].

As an additional term of comparison, in our analysis we also consider an ideal (but unfeasible) scheduling algorithm assuming that: 1) the NC knows, in real time, the precise position of each MN; 2) the new schedule computed by the NC is immediately installed on each MN and BR; and 3) there are no limits on the number of notifications that can be accommodated in a single SU message. Throughout, this ideal algorithm will be referred to as *oracle*.

Below, we compare LASA-R with the existing algorithms presented above (Section VII-A). Then, we evaluate the individual performance of MNs by measuring the PDR achieved by each single MN (Section VII-B). Finally, we investigate the impact of the T_{WAIT} parameter (Section VII-C) and the inaccuracy of the localization system (Section VII-D).

A. Comparison With Existing Algorithms

Fig. 7 shows the PDR obtained with the considered algorithms, in four scenarios characterized by different mobility pattern (linear and random) and speed of MNs ($v = 2$ m/s and $v = 5$ m/s). The results obtained with LASA-R and LASA show that, as expected, in all the considered scenarios the PDR is close to 100% when the number of MNs less or equal to 33. This is because, in these conditions, each MN has a dedicated timeslot in the slotframe and, hence, there are no conflicts.

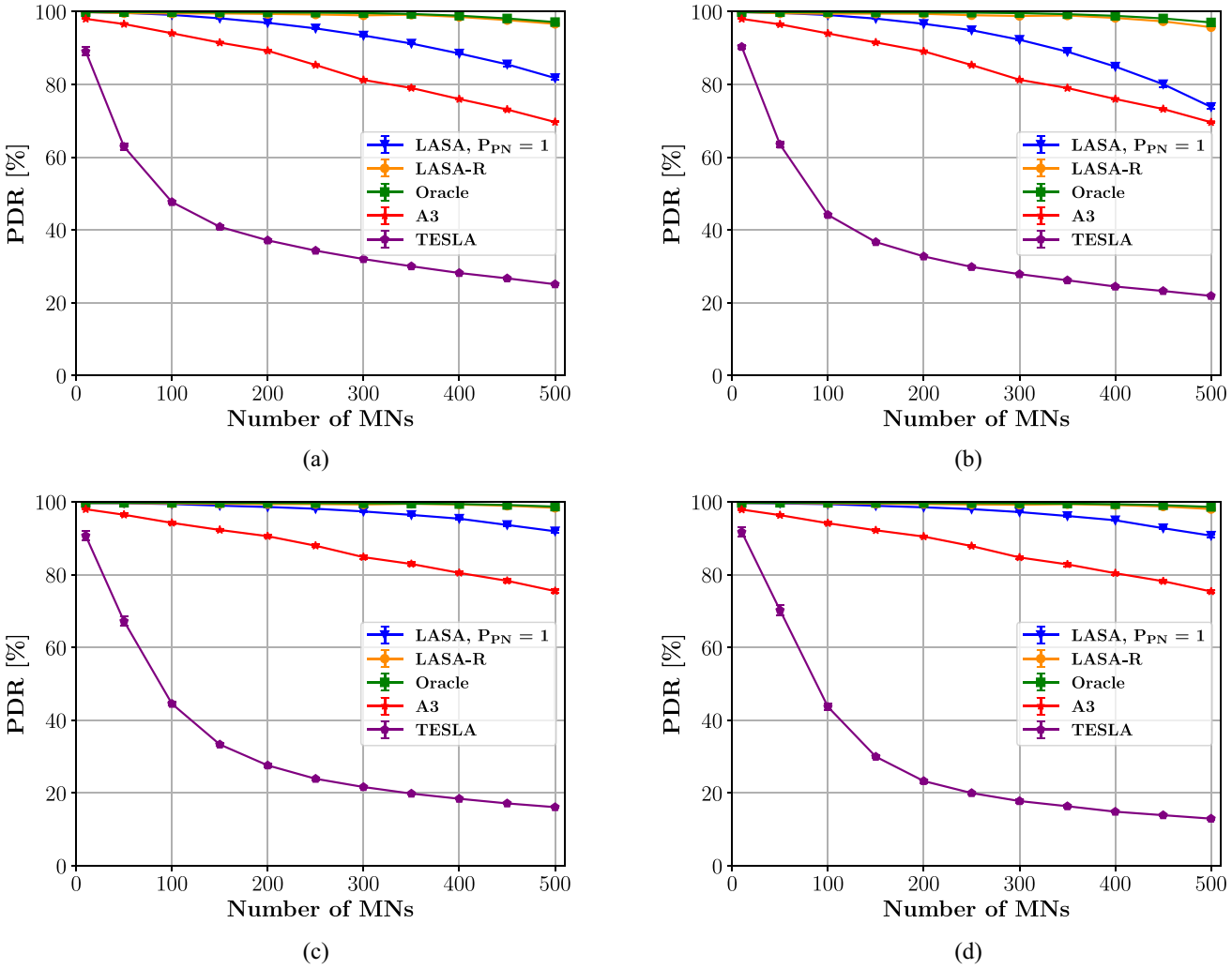

Fig. 7. Comparison between LASA, LASA-R and Oracle, in terms of PDR. (a) Random mobility, v = 2 m/s. (b) Random mobility, v = 5 m/s. (c) Linear mobility, v = 2 m/s. (d) Linear mobility, v = 5 m/s.

As the number of MNs increases, the performance of LASA quickly degrades. In particular, in the scenario with $v = 2$ m/s and random mobility [Fig. 7(a)], LASA exhibits a PDR below 90% when the number of MNs is greater than 400. Instead, LASA-R maintains a PDR above 97% even when the number of MNs is 500. The difference in performance between LASA and LASA-R is even more apparent when considering a higher speed ($v = 5$ m/s). This very significant improvement is due to the rescheduling and refreshing mechanisms, implemented in LASA-R, that are very effective in attenuating the impact of the conflicts, thus resulting in a higher PDR.

Instead, the performance of A3 and TESLA is significantly lower than that of LASA-R and LASA. Specifically, A3 lacks any mechanism to handle conflicts and, hence, when a conflict arises, a random cell is chosen for receiving a packet. This random-based strategy for handling conflicts reduces significantly the PDR, as also highlighted in [14].

TESLA provides the lowest performance among all the considered algorithms, and a rapid drop can be observed in the PDR, even with a limited number of MNs. This is due to the underlying policy used by Orchestra to allocate cells. Since one cell is used by BRs to serve all the MNs in their range, a high number of collisions is experienced by MNs.

The results in Fig. 7(a) also show that, in all the considered scenarios, the performance of LASA-R is very close to that of the ideal oracle. This confirms also that the SA used by LASA-R is effective in selecting the best subset of MNs to be rescheduled and refreshed that can fit in a single SU.

In order to better emphasize the effectiveness of the conflict resolution mechanism of LASA-R, we measured the average fraction of MNs involved in a conflict or whose last transmission was not received by the BR, for each slotframe. For the sake of space, we only show the results for the scenario with random mobility and $v = 5$ m/s (the trend is similar for all the other scenarios). Fig. 8 reports the average percentage of MNs that lost their last data transmission due to a conflict, i.e., MNs that are in a conflict and cannot communicate. The graph shows that, as the number of MNs increases, the average fraction of MNs involved in a conflict increases, as expected. However, while with LASA the fraction grows up to almost 9%, with LASA-R it remains around 2% even with 500 MNs.

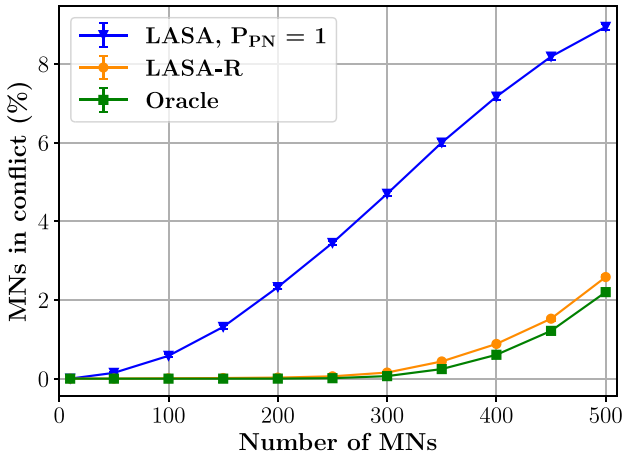

Fig. 8. Percentage of MNs that lost the last data transmission due to a conflict.

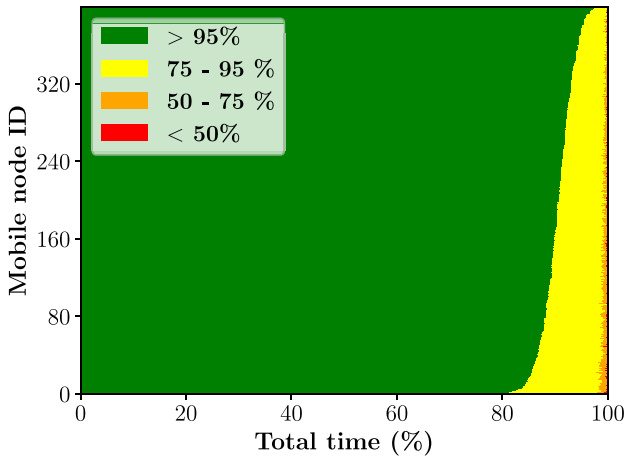

Fig. 9. Individual PDR with LASA-R (400 MNs, randommobility, $v = 5$ m/s).

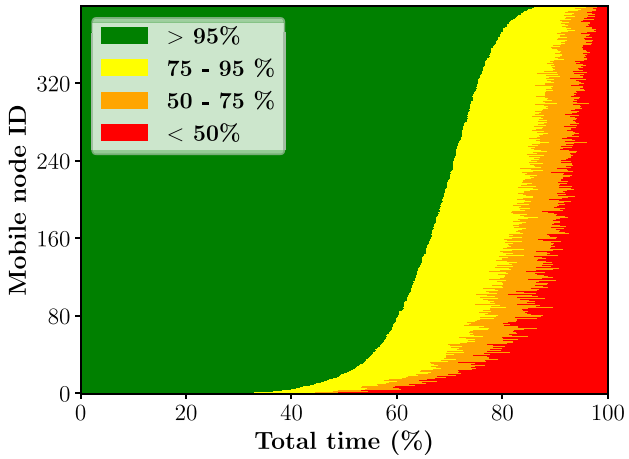

Fig. 10. Individual PDR with LASA (400 MNs, random mobility, $v = 5$ m/s).

Focusing on the impact of the mobility patterns, from the analysis of the Fig. 7, we can observe that the performance gap between LASA-R and LASA is more apparent when the mobility pattern is random. This can be explained as follows. While an MN is involved in a conflict, it cannot communicate and, consequently, it cannot notify its current position to the NC. During this period, the MN can change its position, but the activation/deactivation of cells cannot be performed by BRs. This problem occurs more frequently with LASA, since it does not include a conflict resolution mechanism. In addition, it is exacerbated by random mobility, as MNs can move at larger distances, with respect to the case of linear mobility (a similar effect also occurs when the speed increases). Thanks to the conflict resolution mechanism, this effect is much less relevant in LASA-R and, hence, the impact on performance of mobility pattern and speed is quite limited.

B. Individual Performance of MNs

In order to analyze the individual performance of MNs and assess the fairness of LASA-R, we compare the PDR achieved by each single MN in this section. To this aim, we report the percentage of time (in a simulation experiment) during which an MN experienced a certain PDR, namely, PDR higher than 95%, between 75% and 95%, between 50% and 75%, and lower than 50%.

Ideally, all the MNs should experience the same individual PDR, corresponding to the global PDR. Fig. 9 shows the PDR experienced in practice by each MN with LASA-R. For the sake of space we consider a single experiment with 400 MNs moving with a speed $v = 5$ m/s and random mobility (we assume $T_{WAIT} = 1$). For the sake of readability, the identifiers of MNs are rearranged, based on the experienced PDR, from the highest to the lowest.

The results show that there is a very low fraction of time during which the individual PDR of any MN is below 75%. Instead, for all the MNs, the PDR is greater than 95% for at least 80% of the time. And, if we compare the results of single MNs, we can conclude that LASA-R also ensures a good level of fairness.

For comparison, Fig. 10 shows the corresponding results obtained with LASA. Now, any MN experiences a PDR below 50% for at least 10% of the time. Also, the gap between the best and worst performance is significantly larger than before. This comparison confirms the effectiveness of the LASA-R algorithm, which ensures high communication reliability, fairness, and service continuity.

C. Impact of the T_{WAIT} Parameter

In order to assess the impact of the T_{WAIT} parameter, we performed an additional set of simulations with different values, namely, 1, 4, 8, 16, and 32. For the sake of space, we only show the scenario with random mobility and $v = 5$ m/s, which is the most challenging scenario for LASA-R (the behavior is similar in the other scenarios).

Fig. 11 reports the PDR for different values of T_{WAIT}, with an increasing number of MNs. The impact of T_{WAIT} on the PDR is small. Only a slight decrease in the PDR can be observed, when a large value of T_{WAIT} is adopted, e.g., 16 and 32. This is due to the fact that, even if a large value of T_{WAIT} is set, only those MNs that lost the SU packet are affected. This is a small fraction of the total number of MNs, e.g., about 1 MN per slotframe in the scenario with 400 MNs.

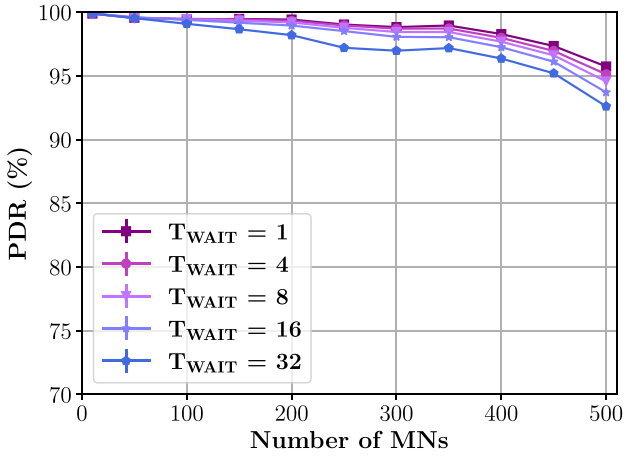

Fig. 11. Impact of TWAIT in LASA-R (random mobility, v =5 m/s).

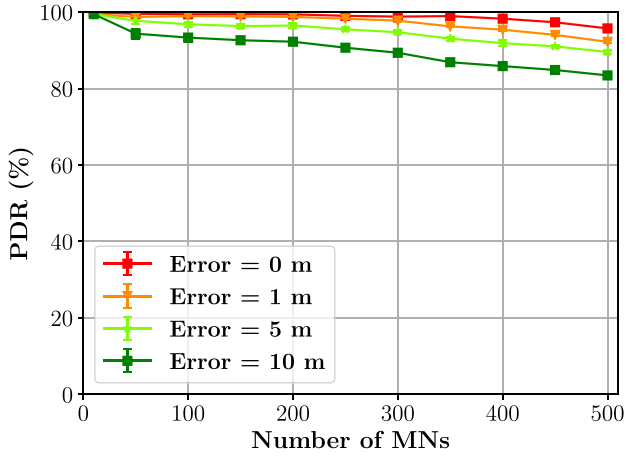

Fig. 12. Impact of inaccuracy in the MN localization (random mobility, $v = 5$ m/s).

Even though the impact of T_{WAIT} is limited, these results suggest that the T_{WAIT} parameter should be set to 1, in order to obtain a very high PDR for all the MNs. A low T_{WAIT} value forces the SA to refresh the MNs that lost the SU packet immediately, thus ensuring a timely recover for MNs that have lost their synchronization.

D. Impact of the Localization System

Finally, we investigate the impact of the localization system, and its inaccuracy in determining the MN position, on the performance achieved by MNs. To this end, we considered three different indoor positioning systems, namely, GPS, UWB, and RSSI-based triangulation. Based on previous measurement studies, the average localization error is roughly 10 m for GPS [23], 5 m for RSSI [24], and 1 m for UWB [25]. Hence, we carried out additional experiments in which the position of MNs was displaced by a distance corresponding to the considered average error (i.e., 1, 5, and 10 m) in a randomly selected direction, with uniform distribution. The results obtained are shown in Fig. 12. As expected, the performance degrades as the average localization error increases. However, when the average error is limited (e.g., 1–2 m), the performance is very close to the ideal case, also with a very large number of MNs. And, even with an average localization error of 10 m, the PDR still remains above 90% with a number of MNs up to 250.

VIII. Conclusion and Future Work

In this article, we have presented the LASA-R for managing mobility in Industrial IoT networks based on the SHMG framework. LASA-R exploits the position of MNs to optimize the allocation of communication resources, while guaranteeing the continuity of service. In addition, LASA-R includes a conflict resolution mechanism that helps in detecting and resolving conflicts, at run time. Finally, LASA-R includes a mathematical framework to derive (offline) the optimal frequency for notifying the position of MNs, thus reaching the best compromise between accuracy and resource consumption.

LASA-R has been evaluated through simulations. Our results show that LASA-R is able to provide continuity of service with a reliability very close to 100%, even when the number of MNs is very large. This is due to the efficient allocation policy and the conflict resolution mechanism. LASA-R also guarantees a low and predictable latency, since each MN uses a dedicated cell for communication to the BR, and BRs are assumed to be connected to the NC through a high-capacity and reliable wired network. Hence, LASA-R is very suitable to Industrial IoT use cases, where applications typically require service continuity, high reliability, and low latency.

In LASA-R, the position of each MN is periodically notified by the MN itself. As a future, we plan to derive a predictive mobility model to forecast the position of the MNs, to further reduce the number of updates exchanged between MNs and the NC and further improve the accuracy of the scheduling algorithm. Additionally, we aspire to enhance conflict management by leveraging available link quality information for each BR and all points within the deployment area.

References

[1] P. Thubert, "An architecture for IPv6 over the time-slotted channel hopping mode of IEEE 802.15.4 (6TiSCH)," Internet Eng. Task Force, RFC 9030, May 2021.
[2] *IEEE Standard for Low-Rate Wireless Networks*, Standard 802.15.4-2020, (Revision of IEEE Std 802.15.4-2015), 2020.
[3] Z. Ming and M. Xu, "NBA: A name-based approach to device mobility in Industrial IoT networks," *Comput. Netw.*, vol. 191, May 2021, Art. no. 107973.
[4] J. Haxhibeqiri, A. Karaağaç, I. Moerman, and J. Hoebeke, "Seamless roaming and guaranteed communication using a synchronized single-hop multi-gateway 802.15.4e TSCH network," *Ad Hoc Netw.*, vol. 86, pp. 1–14, Apr. 2019.
[5] H. Farag, P. Österberg, M. Gidlund, and S. Han, "RMA-RP: A reliable mobility-aware routing protocol for Industrial IoT networks," in *Proc. IEEE Global Conf. Internet Things (GCIoT)*, 2019, pp. 1–6.
[6] O. Tavallaie, J. Taheri, and A. Y. Zomaya, "Design and optimization of traffic-aware TSCH scheduling for mobile 6TiSCH networks," in *Proc. Int. Conf. Internet Things Design Implement.*, 2021, pp. 234–246.
[7] M.-J. Kim and S.-H. Chung, "Efficient route management method for mobile nodes in 6TiSCH network," *Sensors*, vol. 21, no. 9, p. 3074, 2021.
[8] W. Jerbi, O. Cheickhrouhou, A. Guermazi, and H. Trabelsi, "MSU-TSCH: A mobile scheduling updated algorithm for TSCH in the Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 7, pp. 7978–7985, Jul. 2023.
[9] O. Nielsen, L. K. Schnügger, C. Orfanidis, and X. Fafoutis, "Mobility-focused joining in TSCH networks," in *Proc. IEEE 47th Conf. Local Comput. Netw. (LCN)*, 2022, pp. 271–274.
[10] Y. Al-Nidawi and A. H. Kemp, "Mobility aware framework for Timeslotted channel hopping IEEE 802.15.4e sensor networks," *IEEE Sensors J.*, vol. 15, no. 12, pp. 7112–7125, Dec. 2015.

[11] H. Kim, H.-S. Kim, and S. Bahk, "MobiRPL: Adaptive, robust, and RSSI-based mobile routing in low power and lossy networks," *J. Commun. Netw.*, vol. 24, no. 3, pp. 365–383, Jun. 2022.
[12] A. Vaezian and Y. Darmani, "MSE-RPL: Mobility support enhancement in RPL for IoT mobile applications," *IEEE Access*, vol. 10, pp. 80816–80832, 2022.
[13] A. Mohammadsalehi, B. Safaei, A. M. H. Monazzah, L. Bauer, J. Henkel, and A. Ejlali, "ARMOR: A reliable and mobility-aware RPL for mobile Internet of Things infrastructures," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1503–1516, Jan. 2022.
[14] M. Pettorali, F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "LASA: Location-aware scheduling algorithm in Industrial IoT networks with mobile nodes," in *Proc. IEEE 24th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Boston, MA, USA, 2023, pp. 185–194.
[15] M. Pettorali, F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, "Mobility management in industrial IoT environments," in *Proc. IEEE 23rd Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, 2022, pp. 271–280.
[16] S. Kim, H.-S. Kim, and C.-K. Kim, "A3: Adaptive autonomous allocation of TSCH slots," in *Proc. 20th Int. Conf. Inf. Process. Sensor Netw. (CPS)*, 2021, pp. 299–314.
[17] 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.
[18] Q. Wang, X. Vilajosana, and T. Watteyne, "6TiSCH operation Sublayer (6top) protocol (6P)," Internet Eng. Task Force, RFC 8480, Nov. 2018.
[19] R. Alexander et al., "RPL: IPv6 routing protocol for low-power and lossy networks," Internet Eng. Task Force, RFC 6550, Mar. 2012.
[20] M. Pettorali, F. Righetti, and C. Vallati, "Mobile6TiSCH: A simulator for 6TiSCH-based Industrial IoT networks with mobile nodes," in *Proc. 18th Int. Conf. Mobility, Sens. Netw. (MSN)*, 2022, pp. 614–618.
[21] 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.*, 2019, pp. 121–132.
[22] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. 13th ACM Conf. Embed. Netw. Sens. Syst.*, 2015, pp. 337–350.
[23] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk, "Indoor positioning using GPS revisited," in *Proc. Int. Conf. Pervas. Comput.*, Berlin, Germany, 2010, pp. 38–56.
[24] E. Goldoni, A. Savioli, M. Risi, and P. Gamba, "Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4," in *Proc. Eur. Wireless Conf. (EW)*, 2010, pp. 71–77.
[25] A. Poulose, O. S. Eyobu, M. Kim, and D. S. Han, "Localization error analysis of indoor positioning system based on UWB measurements," in *Proc. 11th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, 2019, pp. 84–88.

Marco Pettorali (Member, IEEE) received the master's (cum laude) degree in computer engineering from the University of Pisa, Pisa, Italy, in 2021, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering.

His research interests include Industrial Internet of Things and WSNs for industrial applications.

Mr. Pettorali has served as the Publicity Chair for the 7th Workshop on Smart Service Systems co-located with IEEE SMARTCOMP 2022.

Francesca Righetti received the bachelor's and master's degrees in computer engineering and the Ph.D. degree in information engineering from the University of Pisa, Pisa, Italy, in 2014, 2017, and 2021, respectively.

She is an Assistant Professor with the Department of Information Engineering, University of Pisa. She took part in several national and international projects. Her research interests include WSNs, Internet of Things, and cloud/fog/edge computing.

Dr. Righetti organized the IEEE International Workshop in Smart Service Systems (SmartSys 2022), co-located with IEEE SMARTCOMP. In addition, she has served in the TPC of international conference and workshops, including IEEE SMARTCOMP, IEEE CCNC, IEEE MSN, and IEEE MELECON.

Carlo Vallati received the M.Sc. degree (magna cum laude) and the Ph.D. degree in computer systems engineering from the University of Pisa, Pisa, Italy, in 2008 and 2012, respectively.

He is an Associate Professor with the Department of Information Engineering, University of Pisa. He is also the Director of the Crosslab on Cloud Computing, Big Data and Cybersecurity funded by the Italian Ministry of Education and Research in the framework of the "Departments of Excellence" program. He has been involved in multiple national and international research projects. He has coauthored more than 100 peer-reviewed papers in international journals and conference proceedings.

Dr. Vallati has served as the TPC Co-Chair of IEEE SMARTCOMP 2020 and the General Vice Chair of PERCOM 2022. He is a member of the EB of *Ad Hoc Networks*, *Journal of Reliable Intelligent Environments*, and *Applied Sciences*. He has served as a program committee member for more than 40 international conferences/workshops and as the Workshop Chair for IEEE IoT-SoS and IEEE SmartSys.

Sajal K. Das (Fellow, IEEE) received the M.Sc. degree in computer science and automation from the Indian Institute of Science, Bengaluru, India, in 1984, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, USA, in 1988.

He is a Curators' Distinguished Professor of Computer Science and the Daniel St. Clair Endowed Chair with the Missouri University of Science and Technology, Rolla, MO, USA, where he was the Chair of Computer Science Department from 2013 to 2017. He has published extensively in high quality journals and refereed conference proceedings. He holds five U.S. patents and coauthored four books. His H-index is 98 with more than 38 400 citations according to Google Scholar. His research interests include wireless and sensor networks, mobile and pervasive computing, cyber–physical systems, IoT, smart environments, machine learning, and cyber security.

Dr. Das is a recipient of 14 Best Paper Awards in prestigious conferences, such as ACM MobiCom and IEEE PerCom, and numerous awards for teaching, mentoring, and research, including the IEEE Computer Society's Technical Achievement Award for pioneering contributions to sensor networks and the University of Missouri System President's Award for Sustained Career Excellence. He is a Distinguished Alum of the Indian Institute of Science, Bengaluru. He is the founding Editor-in-Chief of *Pervasive and Mobile Computing* (Elsevier), and an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, and *ACM Transactions on Sensor Networks*.

Giuseppe Anastasi received the M.Sc. degree in electronics engineering and the Ph.D. degree in computer engineering from the University of Pisa, Pisa, Italy, in 1990 and 1995, respectively.

He is a Professor of Computer Engineering with the University of Pisa, where he was the Head of the Department of Information Engineering (DII) from 2016 to 2020. He is also the Director of the CrossLab for the Digital Transformation, funded by the Italian Ministry of Education and Research (MIUR) in the framework of the "Departments of Excellence" program. He has co-edited two books and published about 170 research papers in the area of computer networking and distributed systems. His publications have received more than 11 000 citations, according to Google Scholar (H-index = 45). His current scientific interests include Internet of Things, and cloud/fog/edge computing, cyber–physical systems, cybersecurity, and quantum Internet.

Dr. Anastasi is currently serving as a Steering Committee Member of the IEEE SMARTCOMP conference. Previously, he served as the Area Editor of *Pervasive and Mobile Computing*, *Computer Communications*, and *Sustainable Computing*. He has also served as a general/program chair of many international conferences.