Received February 21, 2020, accepted March 6, 2020, date of publication March 11, 2020, date of current version April 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980119

An Empirical Survey of Autonomous Scheduling Methods for TSCH

ATIS ELSTS[1], (Member, IEEE), SEOHYANG KIM[2,3], HYUNG-SIN KIM[3], AND CHONGKWON KIM[2,3]
[1]Institute of Electronics and Computer Science, LV-1006 Riga, Latvia
[2]Department of Computer Science and Engineering, Seoul National University, Seoul 08816, South Korea
[3]Department of Data Science, Seoul National University, Seoul 08816, South Korea
[4]Institute of Electronics and Computer Science, Seoul National University, Seoul 08816, South Korea

Corresponding author: Chongkwon Kim (ckim@snu.ac.kr)

This work was supported by the ERDF Activity 1.1.1.2 "Post-doctoral Research Aid" (No. 1.1.1.2/VIAA/2/18/282), and also supported by the Institute for Industrial System Innovation and the Institute of Engineering Research at Seoul National University, and the NRF grant funded by the Korean Government (MSIP) (No. 2016R1A5A1012966).

ABSTRACT Time Slotted Channel Hopping (TSCH) is a link layer protocol defined in the IEEE 802.15.4 standard. Although it is designed to provide highly reliable and efficient service targeting industrial automation systems, scheduling TSCH transmissions in the time and frequency dimensions is left to the implementers. We evaluate the performance of existing autonomous scheduling approaches for TSCH on various traffic patterns and network configurations. We thoroughly investigate the pros and cons of each scheme; moreover, we propose the use of node based channel allocation to improve the performance of the best scheme, and demonstrate its practicality and reliability, with up to 6 percentage points better packet delivery ratio than the second best option while retaining a similar radio duty cycle. Finally, based on our extensive performance evaluation, we provide some guidelines on how to select a scheduler for a given network.

INDEX TERMS IEEE 802.15.4, TSCH, autonomous scheduling, MAC, low-power wireless protocols.

I. INTRODUCTION

The TSCH protocol from the IEEE 802.15.4-2015 standard [1] brings highly reliable communications to the field of low-power wireless networks. TSCH has found its uses in industrial monitoring [2], smart homes for healthcare [3], [4], smart buildings [5], environmental monitoring [6] and other areas. One aspect not specified in the IEEE 802.15.4 standard is the construction of a TSCH schedule. Hence, this task has attracted a large attention from the research community.

The approaches to TSCH scheduling can be classified in three main groups: centralized, distributed, and autonomous scheduling. With centralized scheduling [2], a central controller node builds the schedule and distributes it to other nodes. Frequently, this controller is placed outside of the low-power network. With distributed scheduling [7], the schedule is constructed through negotiations between neighboring nodes. The attention of this paper is on *autonomous scheduling* – namely, on approach where the schedule is constructed by each node autonomously, typically (but not exclusively!) relying on routing information already present on the node.

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Khalil Afzal.

Since the publication of the Orchestra scheduler in 2015 [8], autonomous scheduling has proven to be a versatile and reliable component of low-power wireless systems. While centralized scheduling can construct maximally efficient schedules, assuming sufficient information on the controller, it has limited scalability and may entail large overhead for information collection and distribution due to the rapid dynamics of low-power wireless links. Distributed scheduling, on the other hand, is vulnerable to the visibility problem: it is difficult for the network operator to monitor and to debug a low-power network in which the decisions are made through intra-node negotiation in a process that is often complex and opaque. Autonomous scheduling is an attractive alternative as it avoids these problems due to its decentralized nature and low complexity. Furthermore, both centralized and distributed schedulers must rely on some autonomous mechanisms in order to bootstrap their more advanced customized schedules, therefore some kind of autonomous cell allocation mechanism is a necessary part of any TSCH network.

Despite the popularity of autonomous schedulers, a comprehensive evaluation of their various pros and cons is missing from the existing literature. Research papers that present new autonomous schedulers typically evaluate them only on a single traffic pattern or even optimize them for a single specific goal. In contrast, the goal of the present paper is to obtain comparative results covering all of the main autonomous scheduling approaches. To this end, we describe and experimentally evaluate[1] the performance of these approaches on three different traffic patterns: data collection, queries, local traffic, and in two different network types: sparse and dense.

This paper makes the following contributions:

- We provide a taxonomy and description of autonomous TSCH scheduling approaches;
- We model the packet reception probability in three different cell allocation strategies (sender based, receiver based, link based) mathematically, and show that the proposed model has a good fit with numerical simulation results.
- We improve upon the performance of the state-of-the-art approach ALICE [9] by using it in conjunction with a simple but effective idea: a node based channel allocation approach;
- We are the first to scientifically describe and comparatively evaluate an existing approach of the Orchestra [8] scheduler: the Receiver Based Non-Storing variant;
- We provide an experimental performance comparison of all these main autonomous scheduling methods;
- We include provisional guidelines to network designers for choosing an autonomous scheduling method.

The body of this paper starts with a literature survey and taxonomy (Section II). Section III provides theoretical results through modeling and numerical simulations. Section IV presents and analyzes experimental results. Section V includes a discussion with some provisional guidelines, and an overview of future research to be done in the area. Finally, Section VI concludes the paper. We acknowledge the FIT IoT-LAB testbed used for the experimental evaluation.

II. LITERATURE SURVEY AND TAXONOMY

A. BACKGROUND ON TIME SLOTTED CHANNEL HOPPING

To support more reliable and efficient link-layer communication, in the early 2016 the IEEE enhanced the 802.15.4 standard [1] with the 802.15.4e draft amendment [10]. Time Slotted Channel Hopping (TSCH) is one mode defined in the updated standard. It utilizes both time synchronization and frequency hopping techniques. Using this link-layer protocol, neighboring nodes can exchange packets on a scheduled time slot and channel. Since multiple packets cannot be exchanged simultaneously at the same channel, the operation of each node should be organized to avoid collision and achieve high reliability and efficiency. To this end, TSCH utilizes two-dimensional schedule table called *slotframe*. Figure 1 shows an example of tree-structured routing topology and its corresponding TSCH slotframe. The x axis of slotframe is *time offset* and y axis is *channel offset*. We call a pair

[1] Available at https://github.com/atiselsts/tsch-scheduling-comparison

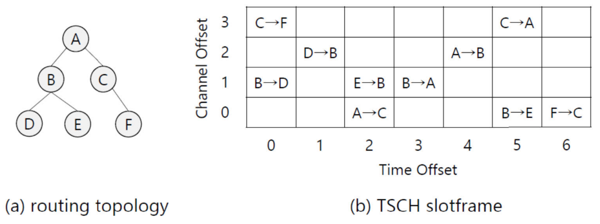


FIGURE 1. Example of a TSCH schedule. a) one example of a multihop routing topology with 6 nodes; b) the corresponding slotframe for TSCH-based communication.

of time offset and channel offset a *cell*; In the example network (Fig. 1), the slotframe has 7 timeslots and 4 channel offsets; the time dimension is also sometimes referred to as the size of the slotframe. Thereby, there are 28 cells in the slotframe. In a cell, a pair of nodes can exchange one data packet and its corresponding acknowledgement. The actions in a cell are done at specific points, according to pre-agreed timing template, such as the template defined in the IEEE 802.15.4 standard. The typical duration of a time slot ranges from 10 ms to 15 ms. In the example network, the node B is scheduled to transmit a packet to the node D at a cell (0,1) and receive packet from the node D at a cell (1,2). The node B is also scheduled to communicate with the node E at the cells (2,1) and (5,0). It is also scheduled to communicate with the node A at the cells (3,1) and (5,3). However, it must be stressed that the IEEE 802.15.4 standard does not define how to schedule the slotframes, leaving it an open research area.

A TSCH network is created by a TSCH coordinator. The coordinator sets some important values such as the network ID, slotframe size, and frequency hopping sequence (FHS), which will be shared by the nodes joined in its network. Then, it creates the network by setting the Absolute Slot Number (ASN) to 0. The ASN increases by one at the end of each timeslot. The coordinator and other joined nodes periodically broadcast Enhanced Beacon (EB) messages, which include important information needed to manage the time synchronized network. Nodes who receive an EB message can join the network by utilizing this information from the EB. When a node joins the network after hearing an EB from its neighbor, it sets that neighbor as its TSCH time source and synchronizes its time to the network time using the ASN. After joining the TSCH network, the node periodically broadcasts EB to make sure its neighbor nodes can join through it and synchronize their time through this node.

A pair of nodes can exchange a data packet and its corresponding acknowledgement in a cell. The transmitter transmits a packet on a specific channel; the receiver must listen on the same channel. Note that the TSCH channel offset does not have one-to-one correspondence with a physical IEEE 802.15.4 channel. Instead, it is as an offset in the FHS. For example, let us assume a TSCH network that uses 15, 11, 19, 13 as its FHS. Then, the FHS of channel offset 0 will be 15, 11, 19, 13, that of channel offset 1 will be 11, 19, 13, 15, and that of channel offset 2 will be 19, 13, 15, 11. Thereby, the channel to be used at *ASN* time slot is:

$$channel = FHS[(ASN + ChannelOffset) \bmod ||FHS||]$$

The scheduled slotframe repeats over time and the effective frequency used by one cell can be changed at the next iteration of the slotframe.

Since packets are forwarded to the final destination on a routing layer, most of the packets are exchanged between routing layer neighbors. Routing Protocol for Low power and Lossy networks (RPL) [11] is the most widely used routing protocol in the current multi-hop networks, and is standardized by the Internet Engineering Task Force (IETF). RPL constructs an directed acyclic graph (DAG) based routing topology. In practice, the DAG is typically implemented as a tree; each node selects one node as its parent, but can have multiple children nodes. RPL supports multiple modes, including the *storing* and *non-storing* modes. In the non-storing mode, each node only keeps the routing entry to its parent. In the storing mode, each node also keeps all routing entries to its children. Thereby, in terms of downward packet transmissions, storing mode can directly support packet exchange between children nodes; however, that comes at the cost of a higher memory usage. In the non-storing mode, all node-to-node routing must take place through the root. For downward packet transmissions, source routing must be used. Generally, TSCH slotframe is scheduled based on the routing topology, so the slotframe should be continuously updated whenever the routing topology changes.

Consider an example: three nodes A, B and C, where node A is the parent of the nodes B and C (Fig. 1), and let us set the slotframe size $M = 7$ and the number of channel offsets to 4. The communication between the nodes should be scheduled on a slotframe. A slotframe can be defined to schedule the node A to listen at the cells (3, 1) and (5, 3), and the nodes B and C to transmit their packets to the node A at the cells (3, 1) and (5, 3), respectively, as shown in Fig. 1. Now, what happens when the routing topology changes and the node B becomes the parent of the node C instead? The nodes A and C remove the scheduled cell for their communication since A is not a parent of C anymore. Instead, nodes B and C add new cells for their communication. As described in this example, the schedule of TSCH slotframe should reflect the relation between neighboring nodes of a routing layer to support efficient multihop communication.

In this paper, we focus on autonomous cell allocation strategies, where the Tx and Rx cells are placed in the slotframes by the nodes themselves, with minimal or no control traffic exchange. Another simple network and a few autonomous schedules for that network are shown in Fig. 2. In a real network, the actual cell placement will likely be different as it is typically decided by a pseudorandom hash function that is shared by all of the nodes in the network; however, the pattern will stay the same. The approaches depicted in the figure are described in the Section II-D.

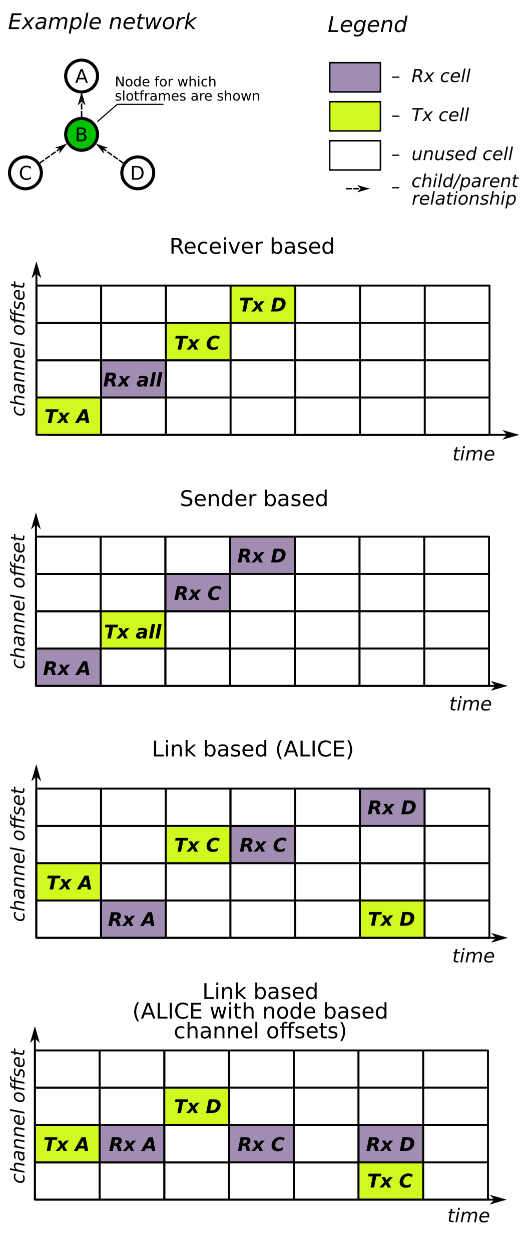


FIGURE 2. Slotframe examples: The figure shows slotframe examples for Node B in the example network using the main autonomous scheduling approaches.

B. SEARCH METHODOLOGY

The literature search was carried out in a systematic way using the SCOPUS database. We searched for the keywords "autonomous" or "autonomously" and "TSCH" in the title, abstract and keyword fields of articles indexed in SCOPUS. This yielded 17 papers published during 2015–2019. The results were then manually filtered to remove irrelevant entries: only English papers where TSCH scheduling was the main topic were included, excluding, for example, papers on TSCH applications such as time synchronization and energy harvesting, and excluding one paper not in English. We also excluded TSCH scheduling papers that were not autonomous, e.g. that relied on 6top [12] or other packet exchange, such as [13], [14]. This filtering yielded 10 papers [8], [9], [15]–[22]. In addition to these search results, we include the Internet Draft of the Minimal Scheduling Function [7] in the survey. Although not a scientific publication, this draft is a strong candidate for an IETF standard, and as such to have an extensive influence on future publications. For similar reasons, we include the 6tisch minimal schedule [23]. Finally, we add one more autonomous scheduling approach: the Orchestra Receiver Based Non-Storing mode (Orchestra RB/NS). We justify this in two ways: first, while Orchestra RB/NS has not been formally described in an individual publication yet, it has already been used in the research literature, e.g., by Mohamadi *et al.* [24] and by Duquennoy *et al.* [25]. Second, it is the default approach in the Contiki[2] and Contiki-NG[3] operating systems, and is the only one that is compatible with non-storing mode routing.

C. TAXONOMY

The literature survey reveals several dimensions on which autonomous scheduling methods can be mapped (Fig. 3). Here we describe these dimensions.

1) INTERACTION WITH ROUTING

If the complete network stack is considered, most of the autonomous scheduling approaches are not fully autonomous. Instead, they just use information from the routing layer to decide which cells to allocate in the schedule. Typically, a node allocates some cells for its parent and children nodes. However, it is also possible to build the schedule completely autonomously, without looking at the routing tree. This is especially useful in the network bootstrap phase, e.g., for neighbor discovery. Such a **routing independent** approach (e.g., Orchestra [8] in the Non-Storing mode) is more general and makes the minimum amount of assumptions; in contrast, the more typical **routing dependent** approaches can be more efficient, as they have access to more information during the schedule allocation time.

2) CELL ALLOCATION

Two broad categories of cell allocation strategies emerge: node based and link based. The node based strategies are further divided in **sender based (SB)** and **receiver based (RB)** [8]. In node based strategies, each node selects a slot that as "its own", typically based on the hash of its MAC address, and schedules a cell in that slot. In a sender based schedule, the node's "own slot" is allocated for transmission (Tx); in receiver based, for reception (Rx) – hence the names of these approaches. Subsequently, the node allocates slots for its neighbor (parent and children) nodes; these are Rx slots in the sender based approach, and Tx slots in the receiver based approach. This means that, paradoxically, most of the active slots in a sender based schedule are for reception, while for receiver based schedule, the opposite is true. In contrast, in a **link based (LB)** approach [9], there are one or more Rx slots and one or more Tx slots for each link. See the Fig. 2 for a visual overview of the resulting slotframes. Assuming a fixed size slotframe and sparse traffic, the receiver based approach leads to lower energy consumption, as it has the minimum amount of Rx slots; however, it is also more susceptible to collisions since all its neighbor nodes are scheduled to transmit packet at the same cell.

[2]http://www.contiki-os.org/
[3]https://www.contiki-ng.org/

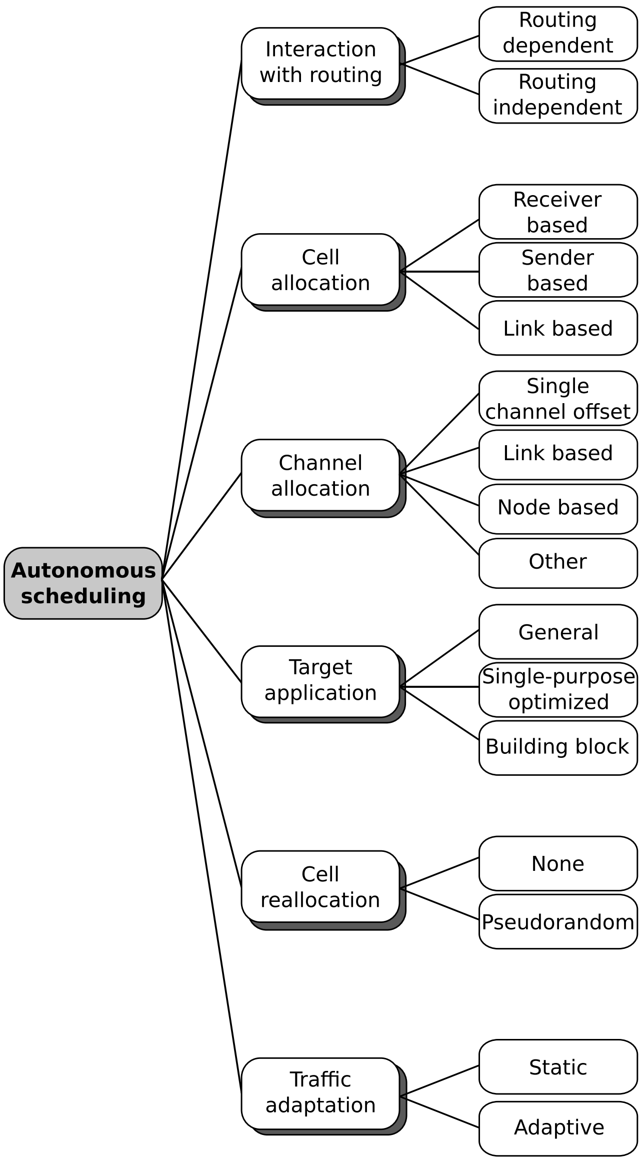


FIGURE 3. Taxonomy of autonomous TSCH scheduling approaches.

3) CHANNEL ALLOCATION

The initial work on autonomous scheduling used a **single channel offset** for all cells [8]. This was demonstrated to be suboptimal [9], [15]. Two other strategies have emerged: **link based** channel allocation [9], possible only in conjunction with link based cell allocation; and **node based** channel allocation [7], [15], [17], [18], where each node selects a single channel offset as ''its own'' and the other nodes accommodate for that. Using multiple channel offsets is beneficial as it allows multiple pairs of nodes in a single radio range to communicate simultaneously without packet collisions utilizing channel diversity. The link based approach has the least amount of collisions; however, it has a different drawback: if a hypothetical node A has Rx cells from both nodes B and C at the same time slot but on the different channel offsets, the node should select only one channel offset it will be activated on since it has only one radio interface. The node randomly selects one of multiple cells since it does not have the information needed to reliably select one; hence, channel mismatch between the sender and receiver may occur. In case of multiple Tx cells in the same timeslot, this problem does not arise. Regarding the mismatch problem, natural tie-breaks exist for cell selection: for example, the node can look at the neighbor queue sizes, and select the cell to the neighbor with the most packets to send [7].

4) TARGET APPLICATION

The main interest of this paper is in **general** autonomous scheduling approaches [8], [9], suitable for data collection, interactive queries and other Internet of Things and sensor network applications. However, some of the surveyed works have been designed to **optimize the performance for a single purpose**, such as low latency [16], or for a specific target application [17]. Furthermore, autonomous scheduling can be a **building block** for more complex scheduling protocols [7], including for network construction and maintenance [21].

5) CELL REALLOCATION

While pseudorandom cell allocation policies lead to good average-case behavior, their worst-case performance can be terrible. With a pseudorandom hash function $h(MAC)$, the worst-case situation is when it maps all MAC addresses in the network to the same timeslot. One way to avoid this is to make the hash function time-dependent. Such a hash function $h(MAC, t)$, even if it leads to a collision in the time moment t_i, is very likely to result in better allocation in the subsequent time moments $t_{i+1}, t_{i+2}, \ldots, t_{i+k-1}$ where k is the period of the hash function. The Absolute Slot Number (ASN) of the TSCH network or the slotframe number (the ASN divided by the slotframe size) can be used as the time parameter t in the hash function. Whenever t changes, the hash function h is recomputed, and all affected cells are reallocated [9]. This is the **pseudorandom** reallocation strategy. A more complex and involved way would be to detect collisions and try to reallocate cells in order to avoid these collisions. This has not been investigated yet, to the best of our knowledge; while MSF includes cell reallocation functionality, it is not applied to the autonomous cells.

6) TRAFFIC ADAPTATION

Purely autonomous approaches do not change the schedule depending on traffic, so they are **static** in this aspect. However, their simplicity can be traded off in order to achieve better performance in networks where the packet rate cannot be reliably known at the time of deployment, or is significantly different on different links or in different periods. Consequently, in **adaptive** approaches [15], [18], [19], the nodes typically start with a purely autonomous schedule, which is subsequently modified and improved through a distributed negotiation process.

D. OVERVIEW OF THE APPROACHES

Table 1 lists the main autonomous scheduling approaches. Note that we treat Orchestra as having node based channel allocation, using multiple channel offsets though the original Orchestra uses a single channel offset for all cells, since this feature is in the version implemented in this paper and released by us in the Contiki-NG 4.4 operating system.

1) GENERAL APPROACHES

The field of autonomous scheduling for TSCH was started by the publication of **Orchestra** by Duquennoy *et al.* in 2015 [8]. Orchestra is an autonomous scheduler that offers several different modes of operation. The original paper describes **receiver based** (RB) and **sender based** (SB) Orchestra modes that both rely on storing routing in order to schedule unicast cells to other nodes (the routing parent and routing children). An unicast cell is also scheduled for the node itself; this cell is Rx in the RB mode and Tx in the SB mode. The cells to other nodes are Tx and Rx, respectively, and scheduled on a cell by hashing its neighbor's address. The timeslot ts_{node} in which to schedule a cell for a node with MAC address MAC_{node} is decided based on Orchestra's hash function $h(MAC)$:

$$ts_{node} := h(MAC_{node}).$$

In the sender based mode, a node always uses its own timeslot to send out packets. In the receiver based mode, upon transmitting a packet to MAC address MAC_{packet}, a node first checks whether a Tx slot for $h(MAC_{packet})$ is in its schedule. If the answer is positive, the timeslot it used; otherwise, the default slotframe (see below) is used. The Tx/Rx cells for the remote nodes are dynamically added in the schedule based on RPL routing information. Whenever a RPL preferred parent is changed or when a route to a direct RPL child is added or removed, the Tx/Rx cells are updated.

In 2016, a new mode was added to Orchestra: **Receiver Based Non-Storing**. This mode is a hybrid between a per-packet scheduler and per-node scheduler. Rx cells are scheduled in same way as in Orchestra RB. In contrast, Tx cells are handled differently: a Tx cell is indiscriminately scheduled on each timeslot. The hash function $h(MAC)$ is used to decide which of those Tx cells to use for which packets. Similarly to the original Orchestra, a packet with destination address MAC_{packet} is scheduled in a unicast timeslot ts_{packet} such that:

$$ts_{packet} := h(MAC_{packet}).$$

Upon transmitting a packet to MAC address MAC_{packet}, a node can always be sure that a Tx timeslot for $h(MAC_{packet})$

TABLE 1. Autonomous TSCH scheduling approaches.

Approach	Routing independent	Cell allocation	Channel allocation	Target application	Cell reallocation	Traffic adaptations
Orchestra SB [8]	–	SB	Node based	General	None	–
Orchestra RB [8]	–	RB	Node based	General	None	–
Orchestra RB Non-Storing (NS)	+	RB	Node based	General	None	–
ALICE [9]	–	LB	Link based	General	Pseudorandom	–
ALICE, node based channel offsets	–	LB	Node based	General	Pseudorandom	–
6tisch Minimal [23]	–	Fixed $ts = 0$	Single	Building block	None	–
Minimal Scheduling Function (MSF) [7]	+	RB	Node based	Building block	None	+
Escalator [16]	–	LB	Custom	Delay optimized	None	–
SPHERE TSCH [17]	–	LB	Node based	Throughput opt.	None	–
TESLA [15]	–	RB	Node based	General	None	+
PAAS [18]	–	Parametrized	Node based	General	None	+
e-TSCH-Orch [19]	–	SB	Single	General	None	+
BOOST [20]	–	Custom	Layer based	General	None	+
Collision-free advertisements [21]	+	Custom	Node based	Adv. traffic	None	–
DiGS [22]	–	SB	Single	General	None	–

is in its schedule. There is no need to update the schedule on RPL changes, and there is no need to ever use the default slotframe. However, more scheduled cells implies a higher memory overhead, as well as higher CPU usage – depending on the implementation, the nodes may have to wake up in every slot.

Besides the unicast slotframe scheduled in the different ways described above, the Orchestra scheduler also includes two other slotframes to exchange control packets: one with slots for Enhanced Beacon (EB) packets, and another with a default timeslot. The default timeslot is shared by all the nodes in the same network. It is used both for broadcast packets (excluding EB) and unicast packets that do not have a matching cell in the unicast slotframe. The EB slotframe is optional; if it is not present in the configuration, EB packets are sent on the default timeslot.

The original Orchestra version puts all cells at the same channel offset; based on the preliminary modeling study in this paper (Section III), we have extended Orchestra with the node based channel allocation approach and report results of this version in the experimental section (Section IV). The node based approach means that each node selects a single channel offset as "its own" for Rx slots and the other nodes accommodate for that when scheduling their Tx slots. Unlike in the case when only a single channel offset is used, different nodes may select different channel offsets for Rx; but unlike in a completely random channel allocation scenario, a single node never uses more than *one* channel offset for its own Rx slots. The modeling study shows that the node based multi channel offset version performs better than the original (single channel offset) Orchestra; this is confirmed by a preliminary experimental study. In this paper, we report just the results from the improved Orchestra.

In 2019, **ALICE** was proposed [9] by Kim *et al.* It departs from Orchestra in three main aspects. First, ALICE uses link based (LB) scheduling instead of node based scheduling. Although link based scheduling has already been used in centralized, distributed, and even some autonomous schedulers [17], ALICE was the first to formulate and investigate this approach in the context of general-purpose autonomous schedulers. Second, ALICE uses link based channel offset allocation instead of using just a single channel offset as in the original Orchestra. Third, ALICE periodically reallocates all unicast cells using a time-dependent hash function. This happens once per slotframe. The authors provide [9] comparative results between ALICE and single-channel offset Orchestra and show that ALICE performs better in all metrics for a data query application. However, its performance in data collection applications has not been investigated so far. The second major limitation of the ALICE study is that they do not attempt to quantify the individual effects from these three main differences. In the present paper, we extend ALICE with node based channel allocation (see the description above), and compare the original version of ALICE with our modification, as well as with Orchestra with node based channel offset allocation (Section IV). We show that most of the benefits in ALICE come from the link based slot allocation. The periodic reallocation does not affect the average-case performance of the network, even though it has a massive effect on the worst-case performance, as some nodes may be

TABLE 2. Qualitative evaluation of autonomous TSCH scheduling approaches. *N* – number of neighbors, *S* – slotframe size, *ST* - subtree size.

Approach	CPU overhead	Memory overhead	Comments
Orchestra SB	$O(N)$	$O(N)$	Advantages: low collisions for data collection; baseline for other approaches.
Orchestra RB	$O(N)$	$O(N)$	Advantages: lower number of Rx slots. Disadvantages: higher number of collisions.
Orchestra RB Non-Storing (NS)	$O(max(N, S))$	$O(S)$	Advantages: routing independent, lower number of Rx slots. Disadvantages: higher CPU and memory overhead, higher number of collisions, bad interplay with transmission backoff.
ALICE	$O(max(N, S))$	$O(N)$	Advantages: lower number of collisions for query applications, collisions are not permanent. Disadv.: higher CPU overhead.
ALICE, node based channel offsets	$O(max(N, S))$	$O(N)$	As in ALICE, except: the receiver never misses packets due to being on wrong channel; more collisions than in ALICE.
6tisch Minimal	$O(1)$	$O(1)$	Advantages: a simple fall-back approach, low CPU and memory overhead. Disadvantages: not efficient.
MSF	$O(N)$	$O(N)$	For autonomous cells: same as for Orchestra RB Non-Storing (NS), but with the advantage of lower CPU and memory overhead due to dynamic cell addition & removal.
Escalator	$O(ST)$	$O(ST)$	Advantages: low delay. Disadvantages: no support for downward traffic, large overhead for nodes on top of big subtrees.
SPHERE TSCH	$O(S)$	$O(S)$	Advantages: high throughput and reliability. Disadvantages: limited number of neighbors, loses efficiency in networks with many hops.
TESLA	$O(N)$	$O(N)$	Advantages: flexibility and traffic load awareness. Disadvantages: high computation overhead and high control traffic overhead.
PAAS	$O(N)$	$O(N)$	Advantages: low collision probability, high throughput. Disadvantages: high energy consumption, high control traffic overhead.
e-TSCH-Orch	$O(N)$	$O(N)$	Advantages: low queue level, simple additional cell allocation. Disadvantages: unstable schedule, high collision probability.
BOOST	$O(S)$	$O(S)$	Advantages: high reliability. Disadvantages: traffic duplication.
Collision-free advertisements	$O(1)$	$O(1)$	Advantages: saves slots due to a multi-subslot approach. Disadvantages: only for advertisements; the protocol depends on having unique IDs pre-assigned to nodes.
DiGS	$O(N)$	$O(N)$	Advantages: increased reliability due to routing path redundancy. Disadvantages: not usable with standard RPL routing.

continuously suffering from a slot collision. Finally, the link based channel offset allocation might cause a cell mismatch problem and actually produces a negative effect compared with the node based channel offset allocation.

The **6tisch minimal** schedule [23] is a very simple scheduling approach that is mainly intended to be used as a way to bootstrap a more complex scheduling scheme. All the nodes in the network share a common cell in a slotframe, and all data packets are exchanged on this cell. Consequently, the 6tisch minimal schedule does not need to generate any control packets. However, the tradeoff here is in the inefficient resource utilization. All transmissions are concentrated on this one cell, therefore the 6tisch minimal schedule does not utilize channel and time diversity. This increases the collision probability due to the lack of communication opportunities. The collision problem may become severe, especially when the traffic load per node increases, and in large & dense networks.

The **Minimal Scheduling Function (MSF)** [7] is defined in an Internet Draft that describes a distributed scheduling mechanism for TSCH on top of the 6top protocol. A full description of the MSF is outside of the scope of this paper. For the purposes of this work, we are interested in the autonomous cells used in the MSF. Each node that runs the MSF allocates some cells autonomously. One Rx cell is allocated for the node itself. Both the time slot and the channel offset are decided used a hash function. The rest of the cells potentially serve as Tx cells. Unlike in Orchestra, the Tx cells in the MSF are added on-needed basis and removed when they are not used anymore, e.g. when there is no traffic to send to a particular neighbor. In case multiple Tx cells are scheduled in the same timeslot, a tie-break mechanism is applied: the cell with the most packets to send is selected. The on-needed aspect only concerns the run-time aspect of the MSF, not the cell allocation method. Considering the evaluation in this paper (Section IV), the MSF is effectively identical to, and expected to achieve the same performance as the Orchestra Receiver Based Non-Storing approach with node based channel offset allocation. The only difference is in the hash function; however, there a no *a priori* reasons why the hash function used by the MSF would make a difference in our evaluation scenarios. Hence, we do not run separate experiments with the MSF autonomous scheduler, expecting identical performance to the Orchestra Receiver Based Non-Storing approach. We hope that our evaluation, especially with the ''local'' traffic pattern, sheds some light on how well the autonomous scheduling component of the MSF is expected to perform relative to other potential approaches. We note that our preliminary experiments investigated the performance of a different older version of the MSF internet draft; the performance of this older version was significantly worse than that of the Orchestra Receiver Based Non-Storing approach.

BOOST [20] is a protocol that blends TSCH with opportunistic routing and autonomous scheduling. BOOST complements the spatial and channel diversity of TSCH with the receiver diversity of opportunistic routing, in this way leading to increased reliability. It supports multiple traffic priorities. Technically, BOOST groups network nodes in layers; the n-th layer consists of nodes n hops away from the root (this information is obtained from the routing protocol). Even-odd transmission/reception slot scheduling is used: depending on the layer, the nodes either transmit or receive in the first slot of the slotframe, do the opposite action in the second slot, and so on. Backoff is used to lower the duty cycle of nodes: if in a Rx slot no packet is received, a subsequent number of Rx slots are skipped. Also, the nodes only wake up in their Tx slots if there are packets to sent.

DiGS (Distributed Graph routing and autonomous Scheduling) [22] is an approach that decentralizes the network management in WirelessHART networks. It allows the field devices to compute their own graph routes and transmission schedules. The authors adopt the WirelessHART graph routing with multiple upstream routes for each node in order to increase reliability by taking advantage of routing path diversity. Multiple transmission attempts are scheduled for each data packet through primary and backup routes. The timeslot for a transmission is decided based on node's ID and the attempt number, effectively creating a variation of the sender based approach.

2) SINGLE-PURPOSE OPTIMIZED APPROACHES

Escalator [16] is a node based scheduling approach which hashes node address to calculate its cell. The authors of Escalator point out that Orchestra does not allocate enough cells for the nodes close to the root even though each node suffers from different traffic load based on the location of the node in a routing tree. To solve the problem, while a node using Orchestra schedules a slotframe by hashing its own address and its neighbor's address, a node using Escalator schedules a slotframe by hashing its own node address and the addresses of all the nodes in its subtree. As a result, a node with a large underlying subtree will have many active cells, thus also will consume a lot more energy. In the best case, Escalator can achieve very low end-to-end latency, since a packet can be forwarded from the source to the destination in the same number of slots as the number of hops between the source and the destination. However, there are no dedicated slots for downward packets, which must be transmitted on the common shared slot, as Escalator focuses just on the data collection scenario.

The **SPHERE** [17], [26] project has deployed a large number of real-life IoT networks in volunteer homes with the aim to monitor residents' health and behavior. IEEE 802.15.4 TSCH is used together with BLE in their networks. To collect a large volume of data from IoT devices, they target high-throughput communication service and use a customized static TSCH schedule to provide both high efficiency and reliability. To this end, the SPHERE TSCH schedule introduces shared slots, the usage of which is negotiated between the parent and children nodes. It also allocates the cells statically, but at the runtime uses the routing state to select one of multiple scheduled cells depending on the node's position in the network, ensuring fast adaptations whenever the routing information changes. In this way, the SPHERE TSCH supports heterogeneous links with high rate unpredictable traffic, and shows high reliability (99.96 % PDR for networks that generate 7.5 packets per second [26]).

The scope of **collision-free advertisement scheduling** [21] is more limited: this work specifically investigates the scheduling of Enhanced Beacon (EB) packets. The paper proposes a novel EB scheduling method and claims that it completely eliminates EB collisions. It is based on two core ideas: first, advertisement slots are divided in subslots, so that in each slot multiple EB packets can be transmitted without collisions. Subsequently, each advertising node is assigned a unique advertisement slot and subslot combination. The schedule depends on all advertising nodes having already assigned unique identifiers that are small integers.

3) ADAPTIVE APPROACHES

Though autonomous TSCH schedulers [8], [9], [16] provide quite reliable service based on simple and efficient algorithms, they fail to provide adaptability on varying traffic load. For example, they just provide fixed number of active slots for each nodes based on given routing topology. The authors of **TESLA** [15] point out to this problem and propose a traffic-aware TSCH cell scheduling method. With TESLA, each node continuously sends its traffic load to its neighbor nodes exchanging control packets. By doing this, all nodes in the network can figure out their neighbors' traffic loads, and add or remove slots from the slotframe continuously. By doing this, they can achieve higher adaptability with enhanced performance in terms of PDR, latency, and radio duty cycle, especially in a high-traffic load scenario. However, as it requires exchange of traffic load information among nodes, TESLA is not fully autonomous anymore - this is a drawback in terms of lightness, simplicity, and speed of the scheduling algorithm.

The authors of **PAAS** [18] also point out that Orchestra does not provide reliable service when a node has many children nodes, as it suffers from high traffic load. Since Orchestra schedules a fixed number of slots on a slotframe not considering collision probability, it suffers from high collision probability on slots on which the traffic is concentrated. By using reserved fields in RPL protocol messages, nodes using PAAS exchange useful information for scheduling and allocates more cells if they are needed. As information is exchanged, PAAS is not a fully-autonomous approach, same as TESLA [15]. However, it achieves higher packet reception ratio and lower latency at the same energy efficiency compared with Orchestra.

e-TSCH-Orch [19] schedules the slotframe based on the Orchestra. The authors of e-TSCH-Orch also points out that Orchestra provides fixed schedule based on the number of routing-layer neighbor. However, each node has different traffic load and some nodes suffer from a number of packets on their transmission queue, which might cause dropping incoming packets. To empty out the transmission queue quickly, e-TSCH-Orch subsequently transmits multiple packets. To this end, a transmitter transmits the number of packets it has in the transmission queue by using TSCH header. The receiver checks the number of packets the transmitter will transmit at a time and immediately schedules the receive cells sequentially. Thereby, the transmitter and the receiver wakes up for the multiple slots exchanging packets. However, without careful priority settings, this approach might ruin the original Orchestra schedule resulting collapse of the network due to severe collision. Moreover, this work assumes packet collection scenario not supporting downward stream scenario.

III. MODELING AND NUMERICAL SIMULATIONS

A. MODELING THE COLLISION PROBABILITY

We are going to use P_{tx} to denote the probability to transmit a packet in an active slot. If this probability is identically and independently distributed, the average number of packets sent in an active slot is given by $k \cdot P_{tx}$, where k is the number of potentially transmitting nodes (i.e., neighbors). There is a collision if and only if $k \cdot P_{tx} > 1$. The probability of collision is zero when $k = 1$, i.e., in sender based or link based slotframes with a collision-free hash function. However, very few hash functions are collision free in real networks.

We can reuse existing results about hash tables to model the collisions for our application. In a M element hash table with A entries filled, the average number of slots with k entries (S_k) is given by:

$$S_k(M, A) = \binom{A}{k} (M-1)^{A-k} / M^A \tag{1}$$

In our application, M is the slotframe size and A the number of active neighbors, i.e., the neighbors sending packets. It then follows that the average number of neighbors sending packets per each active slot (Packets Per Slot, PPS) can be calculated as the weighted average of all S_k where $k \geq 1$.

$$PPS(M, A) = \frac{\sum_{k=1}^{A} k \cdot S_k(M, A)}{\sum_{k=1}^{A} S_k(M, A)} \tag{2}$$

The Figure 4 shows the number of expected number of packets per each active slot. Results from numerical simulations in the figure show a close match with the Eq. 3.

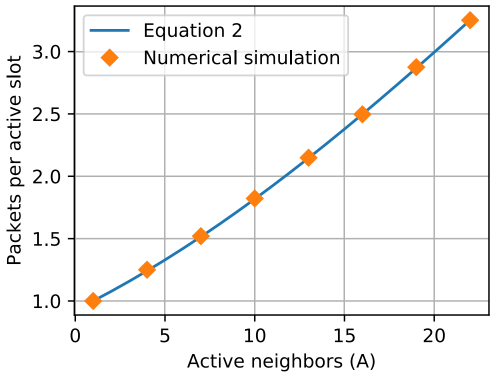


FIGURE 4. The expected number of packets per active slot. $M = 7$.

The Packet Reception Rate of a slotframe (PRR_{SF}) in case of M slots and A active neighbors is the proportion of packets without collisions per slot against all packets per slot. Given that there is a packet in each active cell, the first term is equal to the number of slots with a single active cell, i.e. the value of $S_{k:=1}(M, A)$:

$$PRR_{SF}(M, A) = \frac{S_1(M, A)}{PPS(M, A)} \tag{3}$$

The next challenge is to take into account the transmission probability P_{tx} as normally not all neighbors N are active in each slotframe, but only $P_{tx} \cdot N$ are. The PRR can be expressed as a sum of the different PRR_{SF}, weighted against the probability of seeing each:

$$PRR_{M,N} = \frac{\sum_{A=1}^{N}(\binom{N}{A}P_{tx}^{A}(1-P_{tx})^{(N-A)} \cdot PRR_{SF}(M,A))}{\sum_{A=1}^{N}\binom{N}{A}P_{tx}^{A}(1-P_{tx})^{(N-A)}} \quad (4)$$

The Eq. 4 can now be applied to calculate the upstream PRR of the different cell allocation strategies (Section II-C. With sender based scheduling, the value of PRR (*SBPRR*) is simply the same as Eq. 4, as an Rx cell is allocated randomly in the slotframe for each neighbor:

$$SBPRR_{M,N} = PRR_{M,N} \quad (5)$$

Since we only attempt to analytically model upstream traffic, link based scheduling has the same expected PRR (*LBPRR*) as sender based, as these approaches have the same cell allocation pattern for reception:

$$LBPRR_{M,N} = PRR_{M,N} \quad (6)$$

In contrast, in receiver based scheduling the PRR (*RBPRR*) is expected to be lower, since all neighbors share the same single cell for sending packets. As a result, the slotframe size M is not relevant to *RBPRR*:

$$RBPRR_{M,N} = PRR_{1,N} \quad (7)$$

B. CELL ALLOCATION METHODS

The main cell allocation methods are depicted in Fig. 2. We use our modeling results and numerical simulations to investigate the expected performance of each approach in a simple network. These results are aimed to increase the understanding of each approach and to serve as a motivation for experiments in a real testbed. We look at two different traffic patterns: data collection from leaf nodes, and queries from the central node with replies by the leaf nodes. Further experiments in the testbed (Section IV) additionally investigate the performance of a third pattern: localized parent-child interaction; however, in this single hop network, it is equivalent to the query pattern.

First, we use the Equation 4 to show how P_{tx} affects the different cell allocation approaches. For a data collection application which generates upstream packets only for the application layer service, we can notice that the sender based and link based approaches are identical in terms of the fact that both schemes allocate different slots for each link from the child node to its corresponding parent, hence, only one of them must be modeled. Meanwhile, the receiver based approach allocates only one receive slot from all children nodes to their one common parent which increases collision probability at the common slot. The results (Fig. 5) show the PRR of each scheme when the slotframe size is 7 and the number of nodes is 3. In the graph, the PRR of the receiver based approach decreases faster due to its higher collision probability. This is explained by the fact that in receiver based approach the central node only has a single Rx slot to receive traffic from all leaf nodes; in contrast, with sender and link based methods, it has one to three slots, depending on the results of the pseudorandom hash function that maps between nodes and slots. Furthermore, it is much more likely to have all three slots than just one slot. Through this experiment, we have validated the results of our mathematical model, as we show a good match with the simulation results.

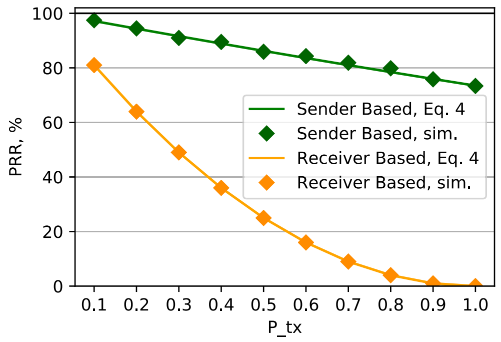


FIGURE 5. **The effect of packet transmission probability depending on the scheduling approach. $M = 7$, $N = 3$.**

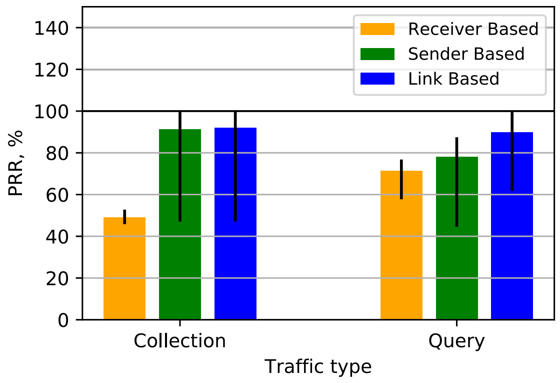


FIGURE 6. **The effect of the scheduling approach. $P_{tx} = 0.3$, $M = 7$, $N = 3$. Numerical simulation results.**

Afterwards we use the now-validated numerical simulations and compare two different traffic patterns: collection and query (Fig. 6). As expected, the receiver based approach leads to worse performance than the other approaches regardless of the traffic pattern. The worst-case performance in data collection (the left side of Fig. 6) is the same for all three approaches, while the average performance of the sender and link based ones is much better, with PRR $> 90\,\%$. With the collection pattern, the performance of the sender based and link based approaches is nearly indistinguishable. In contrast, with the query-type traffic pattern, the link based approach shows much better results than the sender based approach, since there are dedicated Rx cells in both upward and downward directions.

In conclusion, the results in this subsection show that the link based approach is expected to perform better than the other approaches for query traffic. However, for data collection, which is more a general IoT application scenario, both the link based and sender based approaches achieve better performance than the receiver based approach.

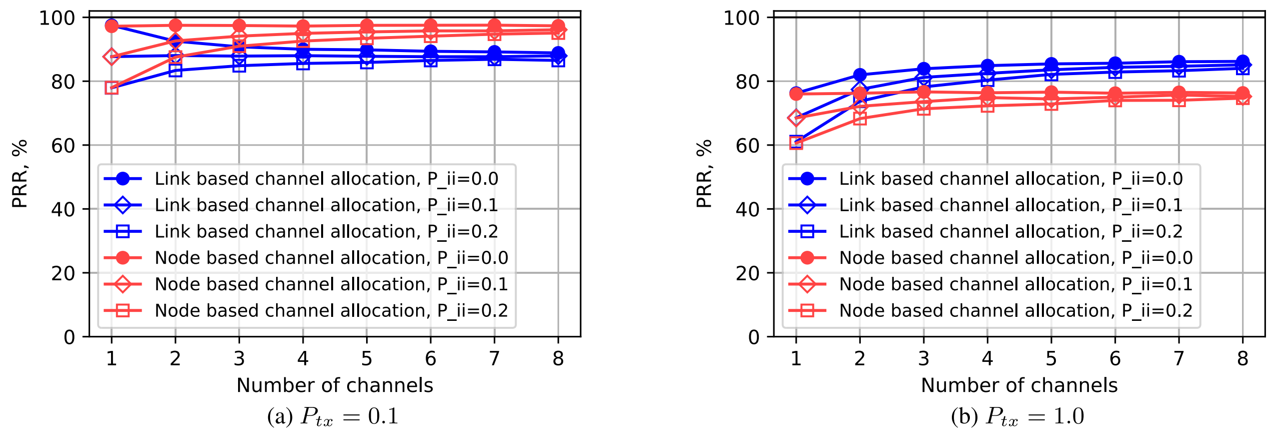


FIGURE 7. **The effect from using multiple channels in a star network with 6 leaf nodes (*N* = 6), with sender or link based slotframe. *M* = 19. Numerical simulation results.**

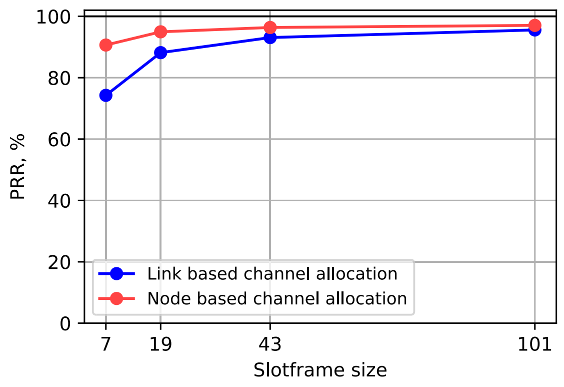


FIGURE 8. **The effect of the slotframe size (*M*) in a star network with 6 leaf nodes (*N* = 6), with sender or link based slotframe. 4 channels, P_{tx} = 0.1, P_{ii} = 0.1. Numerical simulation results.**

C. MULTIPLE CHANNELS AND CHANNEL OFFSETS

In this subsection, we evaluate the effect of the number of channels and channel offsets. Numerical simulation results show (Fig. 7) that the use of multiple physical channels almost always is beneficial. Most of the benefits are already there with as few as 4 active channels, therefore using many more channels than 4 may not be necessary. Multiple channels are additionally useful when there is non-zero internal interference probability ($P_{ii} > 0$) with the rest of a hypothetical network.

Fig. 7a and Fig. 7b show the performance of PRR on low and high traffic load scenarios, respectively. The link based channel offset allocation approach is beneficial for slotframes that have high traffic load (Fig. 7b). With low traffic load (as normally expected on small size networks), node based channel offset allocation shows better results (Figs. 7a, 8). The effect is especially strong for shorter slotframes (Fig. 8). The link based approach is expected to perform particularly good in larger networks and on nodes close to the root, as in tree routing topologies nodes with low routing ranks typically have to forward a high number of packets for their respective sub-trees.

TABLE 3. **Experimental settings.**

Parameter	Value
Operating system	Contiki-NG
Testbed	IoT-LAB, Grenoble
Hardware platform	IoT-LAB M3
Network size	31 nodes
Tx power	3 dBm
Number of channels	4
Unicast SF size	7, 19, 43 and 101 slots
Common SF size	31 slots
EB SF size	397 slots
Routing	RPL (“RPL Classic”)
Data packet interval	6 sec (3 sec for local traffic)
Queue size	8 packets
Warm-up duration	30 minutes
Data gen. duration	30 minutes

The results are only relevant to sender and link based slotframes. With the receiver based approach and in the absence of internal interference, the results would be equal to the results in the Figure 7 when just one channel is used: since all senders transmit at the same timeslot, packet multiplexing via multiple channels would not give any benefits in a star network. Nevertheless, in a multihop network multiple simultaneously communicating pairs of nodes may arise (i.e., $P_{ii} > 0$); this situation would benefit from using multiple channel offsets even with the receiver based approach.

IV. EVALUATION

A. METHODOLOGY

We use the Contiki-NG implementation of TSCH [27] in the FIT IoT-LAB testbed [28] to conduct the experimental evaluation (Table 3).

1) SCHEDULING APPROACH SELECTION

Since our main interest is in the core building blocks of generally-applicable autonomous scheduling mechanisms, we exclude traffic-adaptive and single-metric optimized approaches (Table 1) from the experimental study. That leaves us with the different modes of Orchestra and ALICE. Preliminary investigations showed that our adapted version of Orchestra with multiple channel offsets performs significantly better than the single channel offset version described previously [8], so we only report the results for the former. We do, however, report both the published version of ALICE, with link based channel allocation [9], and our modified one, with node based channel allocation, as we are interested in showing the tradeoffs, as well as the margin of improvement upon the current best state-of-the-art method.

2) TRAFFIC PATTERNS

We investigate the performance on three different traffic patterns:

- **Data collection.** Each node in the network generates a data message and sends it to the root node, resulting in end-to-end upward traffic to the root.
- **Query.** Here, the root generates a query message for each node that is in its routing table. The network nodes reply to the root node, resulting in end-to-end downward traffic from the root and its upward reply to the root.
- **Local traffic.** Each node that has some children generates messages to each child. The child nodes reply to the parent node, resulting in one-hop downward traffic from the parent and its upward reply to the parent.

The first two traffic types emulate typical IoT applications. The third type is designed to mimic the interaction in a distributed scheduling mechanism such as the MSF [7].

3) NETWORK SETTINGS

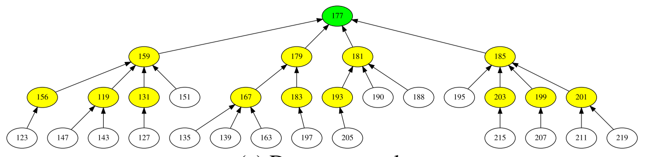
To evaluate performance on various network configurations, we pre-selected some nodes in the IoT-LAB testbed to obtain networks with different densities. First, we pre-selected 31 nodes such that each node has 10 neighbors with link quality ≥50 % on the average generating dense network. Then, we pre-selected another 31 nodes such that each node has only 4 neighbors on the average generating relatively sparse network. As shown in the Fig. 9, these **dense** and **sparse** topologies give rise to considerably different routing trees.

4) METRICS

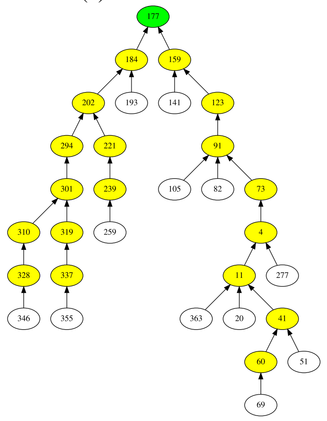
Our evaluation uses the following metrics:

- **Packet Delivery Ratio (PDR).** PDR directly measures the end-to-end reliability of the protocol.
  - For data collection traffic, it is ratio between application-level data packets received on the root against the packets generated on the network nodes.
  - For query traffic, it is ratio between application-level reply packets received on the root against the number of queries issued from the root.
  - For local traffic, it is ratio between application-level reply packets received on parent nodes against the number of queries issued from these nodes.
- **Packet Acknowledgement Ratio (PAR).** PAR is defined for a node as the ratio between the number of MAC-layer packets acknowledged by the parent of the node and the number of packets sent to the parent. Network's PAR is the aggregate of node PAR across all non-root nodes. We do not use Packet Reception Ratio (PRR) here because PAR is easier to track in real networks, and because it shows the effect from link-layer retransmissions.
- **Queue losses.** This is the total number of packets lost due to queue overflow on the sender side. Along with packet collisions and sender/receiver mismatches, it is one of the main sources of packet loss in TSCH networks.
- **Radio duty cycle (RDC).** This is the ratio of the radio-on time against the total time. The RDC in TSCH networks is mostly determined by two factors: the total number of Rx slots, and the number of used Tx slots.

We are interested in the tradeoff between RDC and reliability. The slotframe size cannot be reliably used as a proxy for RDC measurement, since the different approaches add different number of active cells per slotframe. Moreover, different scheduling methods results in different collision probability and different number of retransmissions. Therefore, we measure RDC of each schedulers directly by used slotframe length. Generally, schedulers utilizing more slots achieve



(a) Dense network



(b) Sparse network

FIGURE 9. Routing tree shape depending on experiment configuration settings in the IoT-LAB. Experimental data from representative runs.

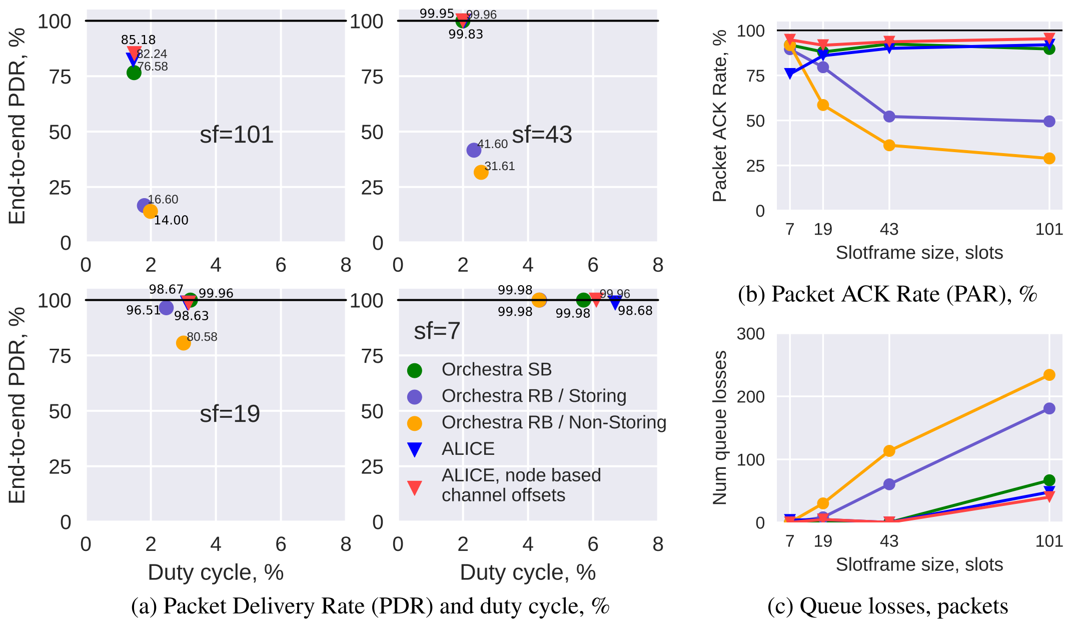


FIGURE 10. **Data collection (from nodes to root), dense network. Experimental results. Here and further: *sf* = <*number*> shows the slotframe size used; the labels of the points show the PDR in percent.**

higher reliability at the cost of high energy consumption. To explicitly show the tradeoff between reliability and energy efficiency, we plot the PDR against RDC.

5) EXPERIMENTAL DETAILS

We report results from a total of 360 hours (15 days) of testbed runs (Table 3), excluding any preliminary investigations. Each option is run for 1 hour, which include 30 min warm-up time with no application traffic for a network bootstrap phase, and 30 min of data packet generation for performance evaluation. The warm-up time is quite long due to the unpredictable time it takes all nodes to join the TSCH network and to discover their neighbors and the link qualities of these neighbors. We shuffle the different protocol options randomly to avoid back-to-back runs of a single protocol. To minimize the impact of random external factors on the results, each option is run three times in total; these runs are then sorted by PDR, and only the results of the median run are reported.

B. EXPERIMENTAL RESULTS

The experimental results are shown in six figures: Fig. 10 and Fig. 11 for data collection, Fig. 12 and Fig. 13 for the query traffic, Fig. 14 and Fig. 15 for the local traffic. In the experiments, we compare the performance of end-to-end PDR and duty cycle of each scheme varying slotframe length from 7 to 101 (Table 3). We also compare the PAR and the number of queue losses by the slotframe length. We repeat the same experiment by varying the network density and traffic pattern.

1) DATA COLLECTION

First of all, we start with the data collection traffic pattern. At the dense network topology (Fig. 10), each node has a larger number of neighbors, thereby ALICE and Orchestra SB schedule more Rx slots per a slotframe. However, Orchestra RB schedules only one Rx slot per a slotframe, regardless of the number of neighbors it has. Thereby, Orchestra RB achieves high PDR only when the slotframe length is extremely low ($sf = 7$); its PDR drastically decreases as the slotframe length increase due to the high number of collisions at the single Rx slot. When the slotframe length is 101, an Orchestra RB node still allocates only one Rx slot per a slotframe; all its neighbors are likely to attempt a packet transmission at this Rx slot, which causes high collisions. However, its duty cycle decreases as the slotframe length increases.

Orchestra SB and ALICE schedule a higher number of cells compared with Orchestra RB, thereby they achieve a higher PDR. ALICE and Orchestra SB show similar performance; ALICE achieves a slightly higher PDR and a lower number of queue losses. When the slotframe is extremely short ($sf = 7$), ALICE shows worse performance compared with Orchestra SB due to the channel mismatch problem. However, by adopting node based channel offsets, we were able to enhance the performance of ALICE; our proposal (ALICE with node based channel offsets) achieves the best performance among all evaluated schemes.

It is interesting to note that the network density clearly affects PDR performance. When the network topology is sparse (Fig. 11), each node has few neighbors, thus a small number of cells per each slotframe are allocated, and opportunities for communication are limited. This results in a long queuing delay and full transmission queues; outgoing packets are dropped. Consequently, as the slotframe length increases, all the schemes suffer from long transmission queues, resulting in queue losses and decreased PDR. In case of Orchestra

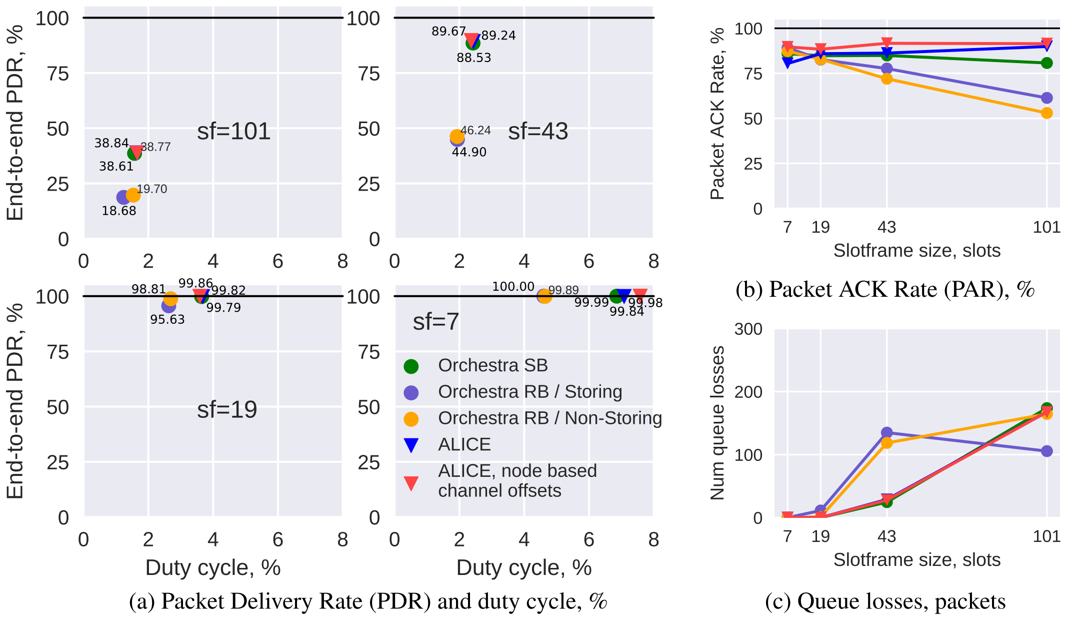


FIGURE 11. Data collection (from nodes to root), sparse network. Experimental results.

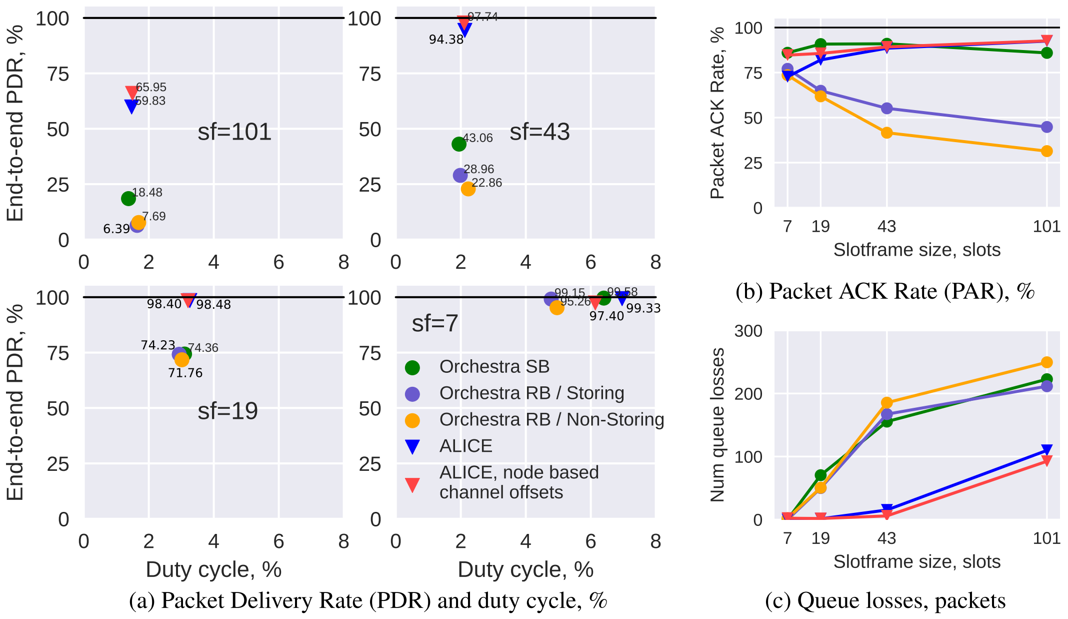


FIGURE 12. Periodic queries (from root to all nodes and back), dense network. Experimental results.

RB, many packets are also lost due to continuous collisions, which result in low PAR and decreased PDR.

2) PERIODIC QUERIES

We repeat the same experiments by changing the traffic pattern from data collection to periodic queries (Fig. 12 and Fig. 13). Here, the root generates downward traffic and each node replies by generating upward traffic, thereby resulting in bidirectional traffic on each link. Since ALICE schedules its slotframes by using directional link based scheduling, it allocates a higher number of cells per slotframe compared with both Orchestra SB and RB; thereby ALICE achieves a higher PDR and lower number of queue losses. Similarly to the previous experiment, the density of the network affects its performance. There is a tradeoff here: in a dense network, each node has a higher number of neighbors and suffers from higher collision and interference probability. This increases the length of transmission queues and the number of packet losses. However, in a dense network a higher number of good

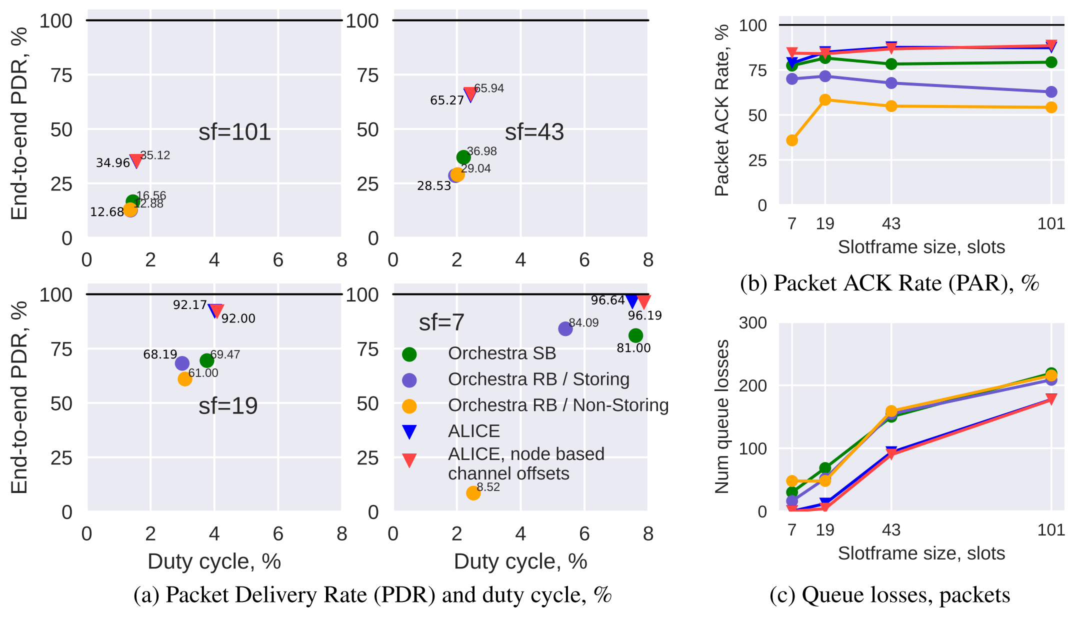


FIGURE 13. Periodic queries (from root to all nodes and back), sparse network. Experimental results.

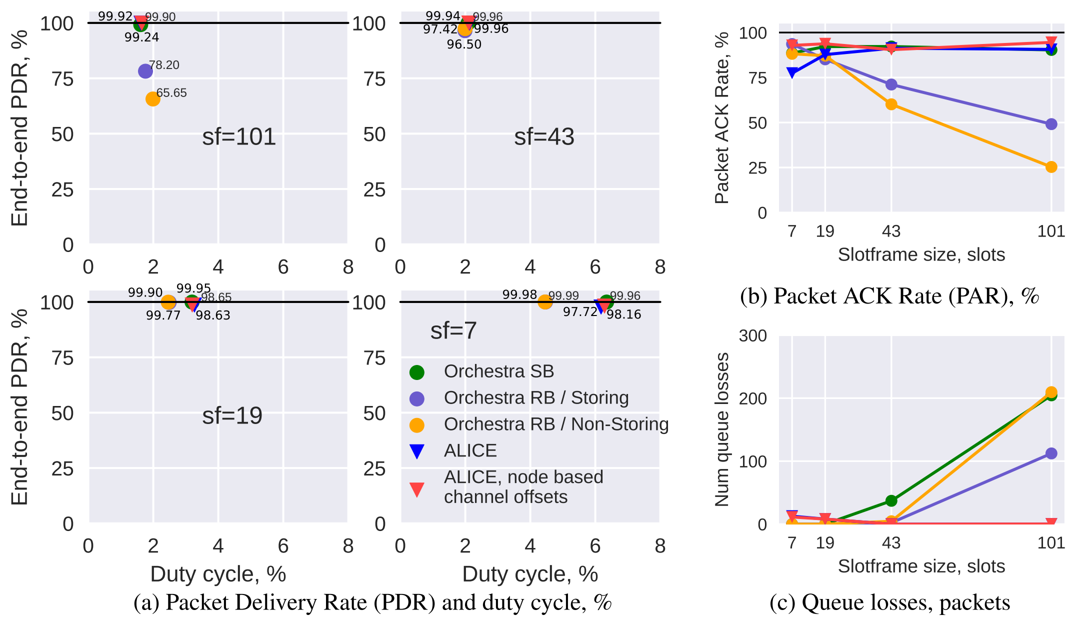
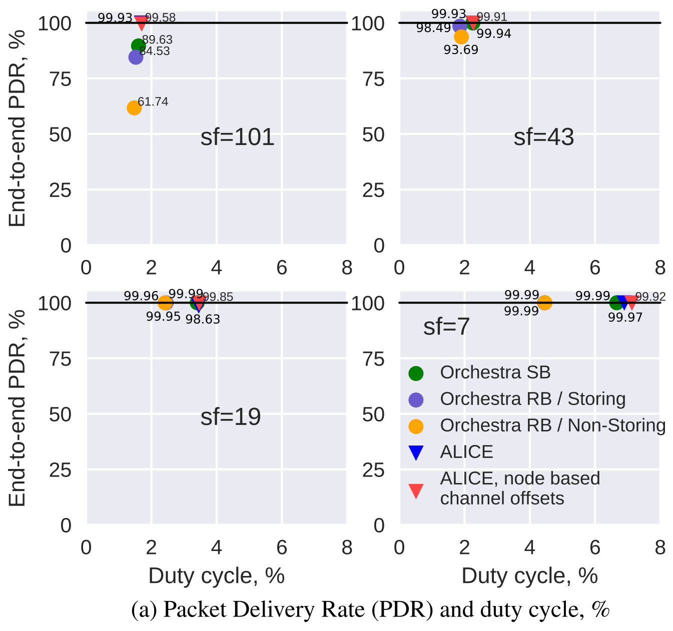


FIGURE 14. Local traffic (node-parent interaction), dense network. Experimental results.

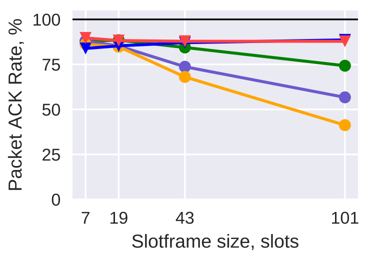
links are available for each node, as well as more choice in terms of the routing parent, and the network has fewer hops. In our experiments, the dense network topology shows better results.

3) LOCAL TRAFFIC

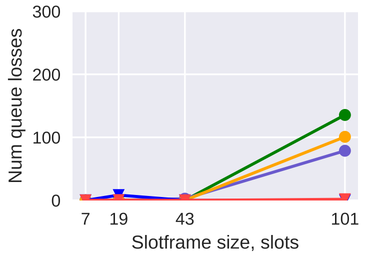
We repeat the same experiments with a local traffic scenario (Fig. 14 and Fig. 15). Here, all application level packets are exchanged between one-hop neighbors. Thereby, the traffic volume is decreased compared with the previous experiments, especially for nodes closer to the root. As a result, each scheduling method achieves increased PDR compared with the previous experiments, but PAR still remains low in case of both Orchestra SB and RB with *sf* = 101. The reason is that both of these Orchestra node based schedulers allocate an insufficient number of slots and thus have insufficient communication opportunities. In contrast, ALICE shows extremely high PAR even with long slotframes due to its increased number of active slots and



(a) Packet Delivery Rate (PDR) and duty cycle, %



(b) Packet ACK Rate (PAR), %



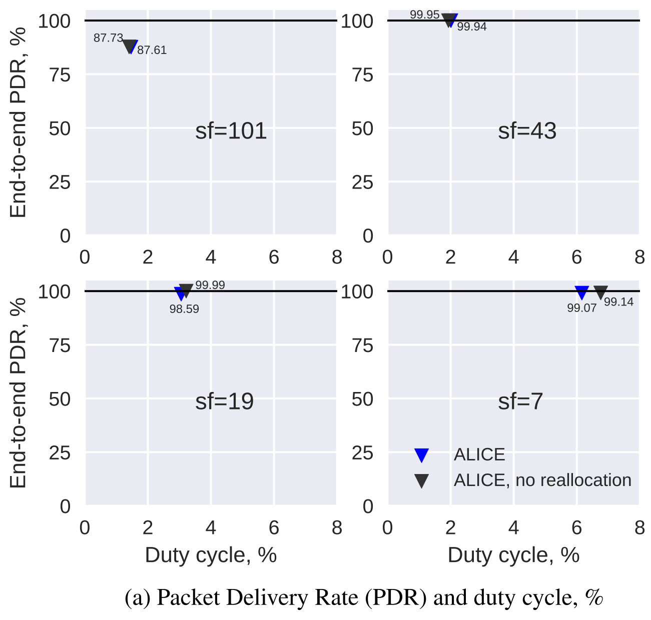
(c) Queue losses, packets

FIGURE 15. **Local traffic (node-parent interaction), sparse network. Experimental results.**

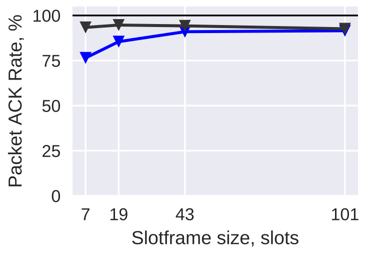
communication opportunities. Its queue losses are extremely low regardless of the slotframe length and network density. Our proposed node based channel offset allocation method for ALICE once again shows its effectiveness and efficiency by demonstrating increased PDR and PAR while keeping the queue losses low.

4) SUMMARY OF THE MAIN RESULTS

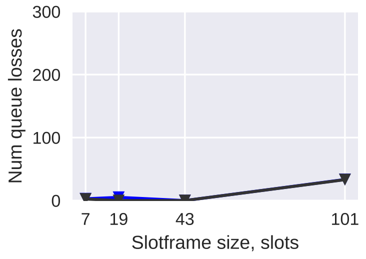
- Receiver based approaches in general perform worse. This holds for all three traffic types, and is consistent with the results in the Section III, Fig. 6.
- Longer slotframes in general correspond to better energy efficiency; however, this becomes less true when the PDR falls a lot below 100 %; see e.g., Orchestra RB in Fig. 13 and Fig. 14.
- The ALICE with a node based channel offset allocation approach dominates all other approaches (within the margin of experimental error) in the query experiments, with both network types, and in the collection experiments, with dense network. This is as expected from the results in the Section III, Fig. 6.
- In the collection experiments with sparse networks, both ALICE options and the Orchestra SB approach show approximately equal results. Still, the ALICE options have better PAR than the sender based approach when the slotframe size is large, and the ALICE node based approach also has a better PAR with the slotframe size is short.
- In all-but-one experiments with short slotframes ALICE with link based channel offsets has the worst PAR of all approaches. Since queue losses with short slotframes are negligible, and as ALICE is expected to have the least amount of collisions from all approaches, the cause of this loss is clear: it is the *channel mismatch* between the sender/receiver cells. That is, the receiver listens to a different channel than the sender transmits to.
- With local traffic, all application level packets are exchanged between one-hop neighbors and not forwarded further, which results in lower network congestion, even for nodes close to the root. There is no clear domination since all approaches show relatively good performance. Still, ALICE has better PAR and fewer queue losses, especially with longer slotframes.
- The storing version of the receiver based Orchestra frequently shows better results than the non-storing version of the receiver based Orchestra. This is explained by implementation differences in the two versions: first of all, the exponential backoff does not work well in the non-storing version; second, the storing version uses the default slotframe more liberally. The exponential backoff increments the "slots backed off" counter on each Tx cell; since the non-storing Orchestra rule adds a Tx cell in each timeslot, the backoff expires very fast and effectively is disabled. In addition, the storing version uses a unicast slot for communication to a node only after a routing relationship has been negotiated with that node through routing protocol message exchange; otherwise, it uses the default slotframe. In contrast, the non-storing option uses a unicast slot to a node whenever the node is present in the neighbor table, and uses the default slotframe only for broadcast packets. This means that the storing option effectively has more Tx slots available for unicast. This becomes especially important when the unicast slotframe is larger than the default slotframe



(a) Packet Delivery Rate (PDR) and duty cycle, %



(b) Packet ACK Rate (PAR), %



(c) Queue losses, packets

FIGURE 16. Effect from the pseudorandom slot reallocation. Data collection, dense network. Experimental results.

(which has size 31). However, the "storing" Orchestra scheduler is somewhat less practical than the "non-storing" one, since the latter can be used in conjunction both with the storing routing, non-storing routing, and in the absence of any routing protocol, while the former only with he storing routing.

5) SLOT REALLOCATION EXPERIMENT

To quantify the effect from the pseudorandom slot reallocation, we run a separate experiment where we compare the default version of ALICE with a "static" version where the pseudorandom slot reallocation is disabled. The Figure 16 shows the results. We fail to detect a significant difference between the two in the PDR and queue loss metrics. As far as the PAR metric is considered, the version without reallocation performs better, indicating a higher likelihood of the sender to receive ACK from the corresponding receiver after the transmission of the data packet in a slot. Due to the calculation time needed for rescheduling the slotframe, it might cause sender/receiver mismatch in terms of slot start time in micro scale. This shows that while there are arguments for slot reallocation due to the worst-case behavior, the impact on average-case performance is negligible, and the slot reallocation functionality cannot be a major reason in the performance differences between ALICE and Orchestra. As a result, we do not include the "static" version of ALICE in the main experiment series.

V. DISCUSSION

A. SELECTION GUIDELINES

Figure 17 shows provisional guidelines for choosing a scheduling approach. For networks that use non-storing routing, there is only one choice: receiver based scheduling. Our experiments show that it has worse performance than the other methods, however, to some extent this can be compensated by the fact that non-storing routing itself performs better than storing routing [25]. The sender based approach never shows significantly better performance than ALICE, consequently, we do not recommend the sender based approach at all. However, in many settings, it shows similar performance, so for existing networks that already use the sender based approach there is no urgent need to upgrade. ALICE with node based channel allocation outperforms ALICE with link based channel allocation (original ALICE [9]) in our experiments, therefore, we recommend ALICE with the link based approach only when the expected utilization of active slots is very high; for the motivation, see the results in Section III-C.

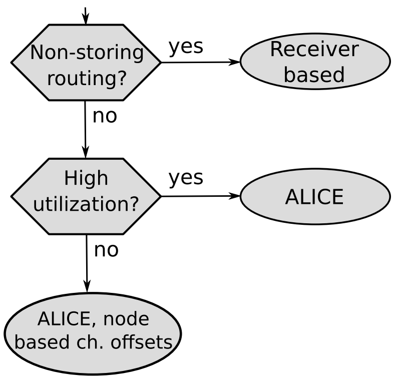


FIGURE 17. Provisional guidelines for choosing a scheduling approach.

Figure 18 gives our recommendations regarding cell reallocation. Our preliminary investigations did not show any significant average-case effect from the reallocation; however,

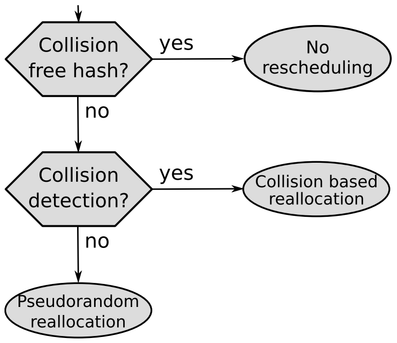


FIGURE 18. Provisional guidelines for choosing a cell reallocation policy.

in a worst-case situation it would make a huge impact on the performance (Section II-C). Reallocation requires some additional CPU time for calculation; however, as the CPU typically has lower energy consumption than the radio, we believe that this cost is justified, and we recommend the reallocation approach for all networks except those where a collision-free hash function for cell allocation is used, or where collision detection is performed.

B. FUTURE WORK

When Table 1 (Section II-D) is considered, it is clear that many combinations in the autonomous scheduler design space (Fig. 3) have not been explored yet, and provide opportunities for future work.

One aspect we noticed in our experiments is that a lot of inefficiency in autonomous scheduling comes down to the fact that the root has to deal with a lot more traffic than the rest of the network. The same is true, to a lesser extent, for other nodes close to the root working as packet forwarders in the multi-hop network. This problem has been considered in MAC design previously, either by allowing for an ''always on'' radio option on the root node, or by developing MAC protocols with behavior dependent on the distance to the root [29]. A simple approach for scheduling would be to introduce a special, ''always Rx'' rule on the root node, and change the schedules of the directly attached nodes respectively. We have already made a tentative investigation of this idea. When it is applied, the performance changes significantly; moreover, the receiver based Orchestra scheduler becomes competitive and in some scenarios even shows the best results of all.

More work towards traffic adaptations is definitely possible and recommended. Some worthy approaches already exist [15], [18]. However, it is a double-edged sword, as going too far in this direction would make the protocol much like another distributed mechanism with all of the associated drawbacks: increased complexity and limited transparency.

In contrast, adaptations to collisions is an idea that has not been explored yet in the context of autonomous scheduling. Similar counterarguments as against traffic adaptations can be made; however, cell reallocation based on collision detection could remove the problem of the terrible worst-case performance, while avoiding pseudorandom reallocation which requires additional CPU load.

VI. CONCLUSION

This paper surveys the state of the art in autonomous scheduling for TSCH. It first walks through the design space of different approaches using mathematical modeling and numerical simulations. Subsequently, for an experimental study we select various modes of Orchestra and ALICE, two general-purpose autonomous schedulers, and present their results on three different traffic patterns and in two different network density settings.

Highlights of the experimental results include:

- We show that receiver based schedulers are not ''more efficient'', as commonly believed. For a constant-size slotframe they do result in lower duty cycle; however, when the tradeoff between PDR and RDC is explicitly considered as a function of slotframe size, other options achieve better PDR at a similar RDC.
- We show that ALICE has much higher PDR than the other options at the same RDC in query applications; however, for other traffic types the sender based Orchestra remains competitive.
- We quantify the average-case effect from the three main features of ALICE, and show that nearly all of its performance gains are due to the link based slot allocation; the pseudorandom reallocation does not have significant average case effect, and the link based slot channel offset allocation has a negative effect unless slot utilization is very high.
- We show that our contribution, ALICE with node based channel allocation, has the best performance of all approaches, and is robust to experimental parameter changes.

Taking into account the experimental results, this paper ends with guidelines on selecting the best-fit autonomous scheduler on different network scenarios.

REFERENCES

[1] *IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Standard 802.15.4-2015, Sep. 2015.

[2] D. Chen, M. Nixon, and A. Mok, *WirelessHART(TM): Real-Time Mesh Network for Industrial Automation*. New York, NY, USA: Springer, 2010, p. 276.

[3] A. Elsts, X. Fafoutis, P. Woznowski, E. Tonkin, G. Oikonomou, R. Piechocki, and I. Craddock, "Enabling healthcare in smart homes: The SPHERE IoT network infrastructure," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 164–170, Dec. 2018.

[4] N. Tsiftes, S. Duquennoy, T. Voigt, M. U. Ahmed, U. Köckemann, and A. Loutfi, "The E-care@ home infrastructure for IoT-enabled healthcare," in *Proc. Int. Conf. IoT Technol. HealthCare*. New York, NY, USA: Springer, 2016, pp. 138–140.

[5] K. Brun-Laguna, A. L. Diedrichs, D. Dujovne, C. Taffernaberry, R. Léone, X. Vilajosana, and T. Watteyne, "Using SmartMesh IP in smart agriculture and smart building applications," *Comput. Commun.*, vol. 121, pp. 83–90, May 2018.

[6] T. Watteyne, A. L. Diedrichs, K. Brun-Laguna, J. E. Chaar, D. Dujovne, J. C. Taffernaberry, and G. Mercado, "PEACH: Predicting frost events in peach orchards using IoT technology," *EAI Endorsed Trans. Internet Things*, vol. 2, no. 5, 2016, Art. no. 151711.

[7] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne, *6TiSCH Minimal Scheduling Function (MSF)*, document, draft-chang-6tisch-msf, IETF, Internet Draft, 2019.

[8] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2015, pp. 337–350.

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

[10] *IEEE Standard for Information Technology, 802.15.4E, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer*, IEEE Standard P802.15.4, IEEE Computer Society, 2012.

[11] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *IPv6 over Low Power Wireless Personal Area Networks (6LowPAN)*, document RFC 4944, IETF, 2007.

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

[13] I. Hosni, "Distributed scheduling with efficient collision detection for end-to-end delay optimization in 6TiSCH multi-hop wireless networks," *Ann. Telecommun.*, vol. 74, nos. 5–6, pp. 239–255, Jun. 2019.

[14] M. V. Ngo, Q. D. La, D. Leong, T. Q. S. Quek, and H. Shin, "User behavior driven MAC scheduling for body sensor networks: A cross-layer approach," *IEEE Sensors J.*, vol. 19, no. 17, pp. 7755–7765, Sep. 2019.

[15] 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.

[16] S. Oh, D. Hwang, K.-H. Kim, and K. Kim, "Escalator: An autonomous scheduling scheme for convergecast in TSCH," *Sensors*, vol. 18, no. 4, p. 1209, 2018.

[17] A. Elsts, X. Fafoutis, J. Pope, G. Oikonomou, R. Piechocki, and I. Craddock, "Scheduling high-rate unpredictable traffic in IEEE 802.15.4 TSCH networks," in *Proc. 13th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Jun. 2017, pp. 3–10.

[18] J. Jung, D. Kim, J. Hong, J. Kang, and Y. Yi, "Parameterized slot scheduling for adaptive and autonomous TSCH networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 76–81.

[19] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. A. Grieco, "Autonomous and traffic-aware scheduling for TSCH networks," *Comput. Netw.*, vol. 135, pp. 201–212, Apr. 2018.

[20] Y. Jin, U. Raza, and M. Sooriyabandara, "BOOST: Bringing opportunistic ROuting and effortless-scheduling to TSCH MAC," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[21] A. Karalis, D. Zorbas, and C. Douligeris, "Collision-free advertisement scheduling for IEEE 802.15.4-TSCH networks," *Sensors*, vol. 19, no. 8, p. 1789, 2019.

[22] J. Shi, M. Sha, and Z. Yang, "DiGS: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 354–364.

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

[24] M. Mohamadi, B. Djamaa, and M. R. Senouci, "Performance evaluation of TSCH-minimal and orchestra scheduling in IEEE 802.15.4e networks," in *Proc. Int. Symp. Program. Syst. (ISPS)*, Apr. 2018, pp. 1–6.

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

[26] A. Elsts, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "TSCH networks for health IoT: Design, evaluation and trials in the wild," *ACM Trans. Internet Things*, vol. 1, no. 1, p. 29, 2020.

[27] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomo, "TSCH and 6TiSCH for contiki: Challenges, design and evaluation," in *Proc. DCOSS*, Jun. 2017, pp. 11–18.

[28] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 459–464.

[29] G.-S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo, "Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks," in *Proc. 4th Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2006, pp. 293–306.

ATIS ELSTS (Member, IEEE) received the Ph.D. degree in computer science from the University of Latvia, in 2014. He was with the Digital Health Engineering Group, University of Bristol, from 2016 to 2018, with the Swedish Institute of Computer Science (SICS), in 2015, and a Researcher with Uppsala University, from 2014 to 2015. Since December 2018, he has been a Researcher with the Institute of Electronics and Computer Science (EDI), Riga, Latvia. He is a maintainer of the Contiki-NG operating system for the Internet of Things (IoT). His scientific interests focus on experimental research in networked embedded Systems, including network protocols, wearable devices, and embedded machine learning.

SEOHYANG KIM received the B.S. degree in computer science from Sejong University, Seoul, South Korea, in 2013, and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, supervised by Prof. Chongkwon Kim, in 2019. Her current research interests include two different research areas: video streaming and sensor network. In video streaming area, she studies client-side ACK regulation for data rate adjustment and balancing the tradeoff between data efficiency and energy efficiency in wireless video streaming. In sensor network area, she studies autonomous TSCH slotframe scheduling method and probabilistic abnormal situation detection.

HYUNG-SIN KIM received the B.S. degree in electrical engineering and the M.S. and Ph.D. degrees in electrical engineering and computer science (EECS) from Seoul National University (SNU), Seoul, South Korea, in 2009, 2011, and 2016, respectively, all with outstanding thesis awards. He was a Postdoctoral Scholar with Network Laboratory (NETLAB), SNU, until August 2016 and Real-time, Intelligent, Secure, Explainable systems (RISELab), University of California, Berkeley, until August 2019, and a Software Engineer with Google Nest, until February 2020. His research interest includes the development of embedded networked systems for the Internet of Things and ambient artificial intelligence. He received the Qualcomm Fellowship, in 2011, and the National Research Foundation (NRF) Global Ph.D. Fellowship and Postdoctoral Fellowship, in 2011 and 2016, respectively.

CHONGKWON KIM received the B.S. degree in industrial engineering from Seoul National University, Seoul, South Korea, the M.S. degree in operations research from the Georgia Institute of Technology, Atlanta, GA, USA, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1981, 1982, and 1987, respectively. In 1987, he joined Bellcore as a member of Technical Staff and worked on Broadband ISDN and ATM. Since 1991, he has been a Professor with the School of Computer Science and Engineering, Seoul National University. His research interests include two different research areas: wireless network and social network. In wireless network, he studies video streaming, sensor network, and MIMO. In social network area, he studies recommender system, spam detection, and graph analysis. He is also interested in high-speed network control, distributed processing, and performance evaluation.