Distributed Slot Scheduling for QoS Guarantee over TSCH-based IoT Networks via Adaptive Parameterization

Jinhwan Jung, Daewoo Kim, Taeyoung Lee, Joohyun Kang, Namjo Ahn, Yung Yi
Department of Electrical Engineering, KAIST, South Korea
{jhjung,dwkim,taeyoung.lee,joohyun.kang,njahn}@lanada.kaist.ac.kr,yiyung@kaist.edu

ABSTRACT

Internet of Things (IoT), which connects a large number of devices with wireless connectivity, has come into the spotlight. As the scope of IoT applications becomes wider, we observe a surge of mission-critical IoT services, e.g., industrial automation systems and medical IoT systems, requiring to satisfy stringent latency, reliability, and/or energy efficiency guarantees. For this purpose, a new MAC, called Time Slotted Channel Hopping (TSCH), has been standardized in IEEE 802.15.4e. However, it is challenging to design a distributed scheduling protocol that achieves the required QoS and energy efficiency at the same time due to complicated tradeoff (providing enough number of slots for QoS vs. minimizing scheduled slots for energy efficiency). In this paper, we propose a novel framework for providing QoS, called SSAP, which is designed to maximize network lifetime in a distributed fashion while satisfying given reliability and latency requirements. To this end, we decompose our goal into two crucial design components: (i) scheduling of slot and channel, and (ii) control of medium access period, each of which is performed by low-complexity and distributed mechanisms. To the best of our knowledge, this paper is the first work to comprehensively handle multiple QoSes for TSCH-based IoT networks. We implement SSAP in Contiki OS and perform extensive simulations and real experiments under various scenarios. Our evaluation results demonstrate that SSAP satisfies highly reliable communication and latency requirements while having the network lifetime that is 1.6 times longer compared to existing protocols for TSCH.

CCS CONCEPTS

• **Networks → Link-layer protocols**; **Network experimentation**;
• **Computer systems organization → Sensor networks**.

KEYWORDS

TSCH, Distributed algorithm, Internet-of-Things (IoT)

1 INTRODUCTION

Among a large class of IoT applications, mission-critical ones, e.g., industrial automation systems and medical IoT systems, which require stringent guarantee of reliability and latency have recently been given considerable attention. To tackle these challenges, a new IoT networking standard, called Time Slotted Channel Hopping (TSCH) [1] has been adopted by IEEE 802.15.4e. TSCH is a TDMA-based MAC protocol with channel hopping, being inherited from the earlier industrial standards such as WirelessHART [30] and ISA100.11a [15]. TSCH is motivated by the fact that contention-based MAC protocols such as CSMA, adopted in many standard and proprietary communication protocols for IoT, may not meet stringent QoS required for mission-critical IoT [32].

One of the core design components for providing QoS in TSCH is a slot/channel (which we call *cell*) scheduling algorithm that determines which slot and which channel to allocate transmitters (TXs) and receivers (RXs). The problem of scheduling slots in TDMA has long been studied over a few decades, where an extensive array of centralized and distributed algorithms have been proposed [9, 14, 29, 31, 33, 34]. Despite many proposals on TDMA slot scheduling, which differs depending on the adopted goal of TSCH, scheduling in TSCH should be designed accordingly, and thus TSCH scheduling protocols with diverse design directions have been proposed, e.g., [2, 5, 6, 8, 10, 13, 17–19, 24, 27, 28, 32]. We are interested in the following key requirements of a good cell scheduling scheme: (i) decentralized operation and (ii) support of high reliability, low latency, and high energy-efficiency, which often have tradeoffs and thus need to strike a good balance among them. An array of recent work on TSCH scheduling has mainly focused on the autonomous operation, largely ignoring the issue of QoS guarantee (see more details in Section 1.1).

In this paper, we propose a distributed scheduling protocol over TSCH that aims at maximizing the network lifetime while satisfying given requirements of latency. In considering the network lifetime, we instill the fairness of lifetime among nodes by employing the notion of α-lifetime, which is analogous to α-fairness [22] in the famous NUM (Network Utility Maximization). As in many other TDMA protocols, each schedule in TSCH is repeated over a collection of consecutive slots, called *Slotframe* or just *frame*. Then, to achieve our goal, we start by considering a problem of jointly finding a cell schedule and an access period schedule. As mentioned earlier, a cell schedule specifies how to allocate transmitters (TXs) and receivers (RXs) to the slots and channels, and an access period of each TX-RX pair e (simply edge), denoted by n_e, which quantifies and parameterizes the interval when an edge e should be activated. For example, $n_e = 3$ means that e is activated every 3 frames. Our protocol regards $\vec{n} = [n_e]_{e \in E}$ as a tunable parameter, which plays a role of decomposing the TSCH scheduling into (i) how to provide at least one time slot for each link and (ii) how often we activate such links. This simple parameterization is very powerful in the sense that each of (i) and (ii) obtains the chance of being easily distributed with low cost operations by forming just a weak-coupling between two components. We now summarize each components as follows:

- ◦ ***Cell scheduling.*** In this component, we just purely focus on developing a cell scheduling which generates almost-zero collisions, i.e., high link-level reliability. To this end, we employ a greedy and distributed cell allocation mechanism to allocate a slot/channel for each edge without overlapping slots and channels among nodes, i.e., providing at least one-slot (ALOS) chance of communication. This cell allocation mechanism can be characterized by the probability to deliver a packet, say r_e, for each edge e. This link-level reliability is transferred to the component of access period control in the form of end-to-end latency.

∘ ***Access period control.*** Given a cell scheduling with the link-level reliability r_e for each edge e, we propose a distributed algorithm that finds the optimal access period vector. By optimal, we mean that it maximizes the network lifetime while satisfying given end-to-end latency on average and traffic requirements. To generalize the network lifetime, we introduce an α-lifetime maximization problem, where using a standard dual-decomposition, we develop a distributed mechanism that requires only local message passing. Our contribution lies in proposing a framework with implementation amenability consisting of ALOS cell scheduling and access period control. Thus, prior cell scheduling algorithms, e.g., [8, 18], can also replace our cell scheduling component with different link-level reliability guarantee, which produces a different optimal access period vector.

Realizing the aforementioned design philosophies, we propose a novel MAC protocol for TSCH with QoS guarantee, called ***SSAP*** (Slot Scheduling with Adaptive Parameterization). To evaluate SSAP, we implement it on the Contiki OS [7]. In simulations, we use Cooja simulator included in the Contiki OS to perform extensive simulations under various controlled environments. To evaluate on a real testbed, we choose IoT-LAB [4] (an open IoT testbed) with 50 M3 nodes where the M3 node is equipped with 2.4 GHz radio for 802.15.4. Our evaluations show that SSAP satisfies the delay requirements from applications while improving the network lifetime, which measures energy efficiency, by up to 60% compared to that of other algorithms.

1.1 Related Work

Prior to TSCH, two TDMA technologies, WirelessHART [30] and ISA100.11a [15], were standardized for industrial applications with many proposals on scheduling algorithms, e.g., [9, 14, 29, 31, 33, 34], just to name a few; see a survey [25] of scheduling algorithms for WirelessHART. Those algorithms are designed to operate with a centralized coordinator that collects information from sensors and disseminates scheduling decisions so as to maximize throughput [9, 34], minimize delay [14, 31] or increase energy efficiency [29, 33].

As an extension of IEEE 802.15.4 [20], which is one of the most widely deployed low power network standards, IEEE 802.15.4e and its MAC framework TSCH [1] have been standardized. In the TSCH standard which inherits WirelessHART [30] and ISA100.11a [15], a list of scheduling algorithms have been proposed [2, 5, 6, 8, 10, 13, 17, 19, 23, 24, 27, 28, 32, 35]; see a survey in [12]. Most of those algorithms are designed to minimize delay and maximize energy efficiency where both the delay and the energy efficiency are closely correlated with how slots are scheduled. We summarize more details of those related work next.

First, there exist researches which propose a centralized (or a distributed but requiring heavy message passing) scheduling method that collects information from IoT sensors, sends the information to the base station, and distributes the determined schedule to each node for delay and energy minimization [2, 27] or for deterministic delay bounds [19]. Second, a series of distributed scheduling algorithms have been proposed [5, 6, 8, 10, 13, 16–18, 24, 26, 28, 32, 36]. Most of them allocate slots for each node, so as to guarantee minimum required slots [6, 10], or minimize delay [5, 13, 24, 32] in a distributed manner. Orchestra [8] was proposed as an autonomous scheduling algorithm, and then the protocols improving Orchestra have been proposed to take the tradeoff between energy efficiency and reliability [17], to minimize delay [28], or to support heavy bi-directional traffic [18].

Difference from prior work. Looking at the recent proposals on TSCH scheduling, e.g., Orchestra [8] and its follow-up works [17, 18, 28], their focus was more on autonomous scheduling methods which allocate slots based only on the routing information without any overhead for scheduling. Those autonomous scheduling protocols would have a big advantage for dynamic environments. However, we focus more on QoS guarantee because emerging industrial IoT applications are becoming QoS-hugry in terms of energy efficiency, latency, and reliability. Note that we consider the latency QoS as the expected end-to-end latency so as to achieve the QoSes of energy efficiency and latency simultaneously in a distributed manner.[1] In handling such issues, the key problem is to construct an easily implementable framework which finds the "optimal" way of scheduling slots/channels and to offer a tunable control knob for choosing a certain vector of multiple QoSes, such that (a) the number of slots allocated in each node is well balanced over a network, (b) the number of slots allocated for each path from a source to a destination is enough to achieve delay requirements on average, and (c) those slots should be scheduled without overlaps for high reliability. Note that those metrics are closely coupled and are often in conflict with each other. In this paper, we propose a parameterized scheduling framework that decomposes the scheduling problem into two components: (i) *distributed cell scheduling with at-least one-slot guarantee*, and (ii) *distributed mechanism for access period control.*

2 BACKGROUND AND DESIGN FRAMEWORK

In this section, we present the background of TSCH with emphasis on slot scheduling in TSCH in Section 2.1, and in Section 2.2 we describe the goal of this paper and the overview of our design.

2.1 Primer on TSCH

A new scheduling-based MAC, called Time Slotted Channel Hopping (TSCH), has been proposed in IEEE 802.15.4e [1] for IoT applications requiring stringent QoS guarantee. Every node in TSCH should transmit Enhanced Beacon (EB) periodically to advertise its network information. By transmitting and receiving packets and ACKs with its time sources (typically its parent node in the tree-based routing structure), each node are globally synchronized. TSCH aims at achieving high reliability and low latency through Time Division Multiple Access (TDMA) with channel hopping. Time is divided into a sequence of slots, where as in other TDMA-based protocols, TSCH has a fixed number of slots, called Slotframe, with a certain scheduling pattern, which is repeated over time. To maximize frequency utilization, there are multiple channels (16 in the standard [20]).

[1] Some prior work [19] provided deterministic delay QoS, but requiring too much message overhead and energy consumption.

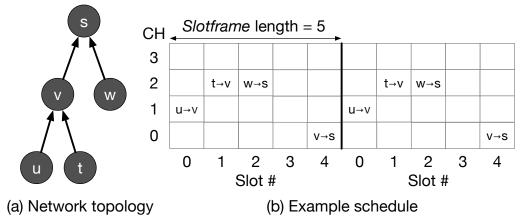


Figure 1: A network topology with 5 nodes and an example cell schedule. Each TX-RX pair is allocated in a different cell (e.g., a pair uv is allocated at $(l_{uv} = 0, c_{uv} = 1)$) where $L_{\mathbf{SF}} = 5$ and $N_{\mathbf{CH}} = 4$. This schedule is repeated over Slotframes.

Of critical importance is *cell scheduling* (or simply scheduling) that determines which (slot, channel) is allocated for each TX-RX pair uv. Each TX-RX pair is allowed to perform channel hopping by being allocated different channels over successive Slotframes. The way of cell scheduling is not specified by TSCH, which is rather vendor-specific. Let L_{SF} be the length of Slotframe and N_{CH} be the number of channels. Then, a cell is a combination of a slot and a channel, and given a TX-RX pair uv, we denote by $S_{uv} = (l_{uv}, c_{uv})$ the cell allocated to uv, where $0 \leq l_{uv} \leq L_{\text{SF}} - 1$ and $0 \leq c_{uv} \leq N_{\text{CH}} - 1$. Figs. 1 show an example network and a schedule of cell allocation with 5 nodes. Each TX-RX pair is allocated at a certain cell. For example, $u \longrightarrow v$ means that u as a TX and v as a RX are allocated at slot #0 and the channel 1.

2.2 Goal and Design Overview

Goal. Our goal is to develop a scheduling algorithm that achieves end-to-end latency requirement on average while minimizing energy consumption and stabilizing the network. We focus on providing the expected latency guarantee rather than strict one. With random arrival of events the strict guarantee requires too many slots (i.e., energy consumption) or even is infeasible due to nondeterministic delays (e.g., queueing, link loss), while the average guarantee can be achieved with Poission arrival, which is practical QoS for IoT applications. Hence, we aim to guarantee the expected latency while maximizing energy efficiency. In terms of energy efficiency, we consider network lifetime, which quantifies how balanced energy consumption is across the network. We aim at achieving our goal by solving the following form of optimization: maximize the network lifetime subject to the constraint of expected end-to-end latency.

Decomposition via weak coupling: Schedule and period control. To this end, we take an approach of studying the target optimization by considering the following two weakly-coupled components. (i) *at-least-one-slot (ALOS) cell scheduling:* constructing a cell schedule that guarantees at least one chance of communication for each TX-RX, and (ii) *access period control:* controlling how often such a schedule is activated (i.e., the period between activation) in a TX-RX pair-specific manner. We note that some of the prior work on TSCH scheduling [8, 18] has already proposed mechanisms with the objective in (i). However, they did not pay much attention to providing QoSes. What we do in this paper is as follows: first, as described in (ii), we smartly control how often each TX-RX pair should be activated by understanding how it tradeoffs reliability, latency, and network lifetime, and second, we propose a new ALOS cell scheduling which is more amenable to access period control in (ii) and thus provides better QoSes. This decomposition with the weak-coupling of two modules facilitates to develop a protocol that operate in a distributed manner. More precisely, ALOS cell scheduling solely focuses on maximizing reliability while access period control mainly trades off latency and energy efficiency. Those different objectives give us a chance to decompose the problem and thus develop the distributed protocol. We note that this decomposition incurs no loss of optimality if we assume Poisson arrival of events with small rate.

An intuitive explanation of how access period control has impact on multiple QoSes is as follows: When a TX-RX pair, say uv, has smaller access period, implying that the cell schedule for uv is activated frequently with the short period, it has a higher chance to communicate packets, so that those packets experience low latency. However, due to frequent activations, the RX v tends to consume more energy by listening more frequently. On the contrary, it is obvious that RX can save more energy with larger access period, resulting in longer delay. Since the end-to-end delay of a packet is determined by access period of the nodes in the routing path of the packet and each node in the network is likely to have different offered traffic load, for longer network lifetime, it is important to balance the energy consumption over the entire network and guarantee the given end-to-end delay by achieving proper per-link delay by controlling access period.

Where is it coupled? Since the goal of scheduling becomes how we provide at-least-one-slot chance of communication, the way it is coupled with access period control is on how much a given ALOS scheduling algorithm guarantees the success of a packet delivery, i.e., reliability. This is because the amount of reliability has an impact on latency for 100% packet delivery, when we assume repeated retransmission until the success of a target packet.[2] This significantly simplifies the design of ALOS cell scheduling, so that its objective is simply reduced to the one with the minimum number of collisions.

Design overview. We now provide an overview of the design of our proposed protocol, called ***SSAP*** (Slot Scheduling with Adaptive Parameterization). SSAP is designed on top of TSCH and RPL (Routing Protocol for Low-Power and Lossy Networks) which is the *de facto* standard routing protocol; thus, we assume a tree structure of nodes formed by RPL. As mentioned earlier, the key idea is to use the weak coupling of cell scheduling and access period control by purely focusing on developing (i) a distributed scheduling with high reliability and (ii) a distributed algorithm to search for the "best" access period parameter that maximizes the network lifetime while guaranteeing the end-to-end latency[3]. This corresponds to the strategy that a cell scheduling algorithm is developed under the assumption of the access period $n_e = 1$, for all $e \in E$, corresponding to the worst-case of high offered load, and then controls access period to adjust itself to actual loads but considers the network lifetime and latency. Note that, as discussed in how two modules are coupled, in achieving the end-to-end latency as controlling access period, we use the per-link latency guarantee provided by the proposed cell scheduling algorithm.

[2]In practice, we limit the maximum number of retransmissions.
[3]This implies the expected end-to-end latency unless otherwise noted.

3 CELL SCHEDULING

In this section, we present cell scheduling module of SSAP. In terms of scheduling, we focus on one of the popular IoT applications of collecting sensing data from sensors to a base station given a tree topology by RPL. Although SSAP considers both uplink and downlink traffic, we only use uplink traffic case to describe the scheduling algorithm for simplicity. Downlink traffic case is discussed at the end of this section.

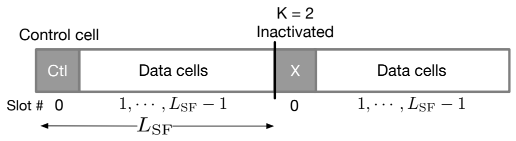


Figure 2: Structure of Slotframe, where a hyperparameter K controls how often the control cell is activated (e.g., $K = 2$).

3.1 Frame Structure

A Slotframe consists of one control cell and $L_{\mathrm{SF}} - 1$ data cells, as depicted in Fig. 2. As done in many TDMA-based MAC protocols, a control cell is used to deliver control information such as Enhanced Beacon (EB) for TSCH network advertisement and RPL control packets for exchanging routing information. Although we mainly focus on the unicast traffic, thus although it is beyond the scope of this paper, it is also possible to use this control cell for any broadcast traffic. We consider the case when control packets' loads are not so significant, so it is enough to activate control cells every K Slotframe.

3.2 Distributed Data Cell Scheduling

We describe our cell scheduling protocol by explaining at which slot and channel an arbitrary link vp is scheduled, where v is a node and p is its parent. Since the routing paths are in the form of a tree and we focus on uplink traffic, the cell S_{vp} of link vp can be simply denoted by $S_v = (l_v, c_v)$ for simplicity, unless confusion arises, where recall that l_v and c_v are the allocated slot and channel for vp, respectively.

Our goal of cell scheduling is to reduce as many conflicts as possible where the conflict denotes either of a collision or interference. We assume that two communications, which are performed over two links in the same slot and channel, say e_1 and e_2, conflict with each other, if they are located within two hops. However, if they occur in different channels, they are conflict-free. This interference model is known to be suitable for unicast communications using ACKs, e.g., Wi-Fi 802.11. We also assume half-duplex radios, implying that the communications over e_1 and e_2 in different channels become successful, as long as e_1 and e_2 are not connected (i.e., located within one hop).

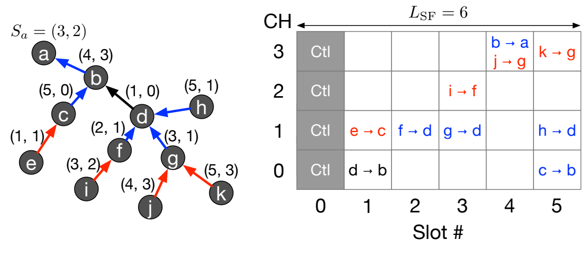
Algorithm description. In this section, we describe how the cell scheduling algorithm allocates data cells in a distributed manner. Based on the given tree structure, our cell scheduling algorithm is performed in a top-down manner. In other words, from a base station (i.e., the root of the tree) to the leaf nodes in the tree. When a node v receives its cell schedule S_v from its parent node p, v schedules all of its child nodes $c \in C_v$ to find a set of cells $\{S_c\}_{c \in C_v}$ such that (i) collisions are avoided by allocating different slots in the time domain, and (ii) interferences are minimized by setting a channel carefully in the frequency domain. Now we present the detailed description of the cell scheduling algorithm.

Algorithm 1: Cell scheduling algorithm from nodes $v \rightarrow c$

```
Input: Cell of $v$: $S_v = (l_v, c_v)$, slot of $p$: $l_p$
Output: Cells of $v$'s child $c \in C_v$: $S_c = (l_c, c_c)$
1  Set list $\mathcal{A}_v := [l_v + 1, l_v + 2, ..., L_{\mathrm{SF}} - 1, 1, ..., l_v - 1]$
2  Initialize $\mathcal{B}_v = []$
3  for child node $c \in C_v$ do
4    while true
5      if $\mathcal{A}_v \neq \varnothing$ then
6        $l_c \leftarrow$ Pop the first element from $\mathcal{A}_v$
7        if $l_c \neq l_p$ then
8          Append $l_c$ at the end of $\mathcal{B}_v$, then break
9      else
10       $l_c \leftarrow$ Pop the first element from $\mathcal{B}_v$
11       Append $l_c$ at the end of $\mathcal{B}_v$, then break
12   $c_c \leftarrow l_v \bmod N_{\mathrm{CH}}$
13   $S_c \leftarrow (l_c, c_c)$
14 return a set of cells $\{S_c\}_{c \in C_v}$
```

Algorithm 1 describes the cell scheduling algorithm in terms of how a node v allocates its child nodes $c \in C_v$. Initially, v sets a list $\mathcal{A}_v$ that contains every slot of Slotframe starting from $l_v + 1$, except l_v for itself and the slot for control cell. Another list $\mathcal{B}_v$ is initialized in which this list stores allocated slots. To prevent collision, v allocates a different slot to each of its child node c by popping the first element from $\mathcal{A}_v$ (Line 6). The reason why l_c should be different from l_p (Line 7) is to avoid 2-hop interference, which is described in detail later. By allocating l_c for all child nodes as described above, those nodes will be spread over the time domain as much as possible if the number of available slots is larger than the number of child nodes. If $\mathcal{A}_v$ becomes empty, a slot should be allocated among already allocated slots (which is maintained in Line 8). By choosing the first element from $\mathcal{B}_v$ and appending it at the end of $\mathcal{B}_v$, this slot will be allocated to one of the already allocated slot, but the number of co-allocated child nodes is minimized (Line 10-11). It is worth noting that those child nodes which are already allocated separately in the time domain can use the same channel without collisions. Thus, v allocates a channel c_c such that $c_c = l_v \bmod N_{\mathrm{CH}}$ where mod is modulo operator and N_{CH} is the number of available channels (Line 12). This channel allocation indicates that when $c \in C_v$ transmits a packet to v, they use the same channel c_c. Since c_c is obtained from l_v, c and some nodes which are apart from 2-hop from c must use different channels if the number of available channels is enough; thus, our cell scheduling minimizes 2-hop interference by separating them in the frequency domain. By repeating the above procedure for all of v's child nodes, v obtains a set of cells $\{S_c\}_{c \in C_v}$ that minimizes the collision and interference as much as possible.

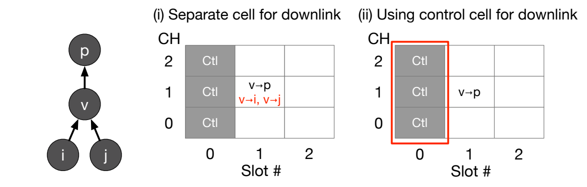
Example. We take an example to explain how the cell scheduling algorithm actually runs. Fig. 3(a) shows a tree with 11 nodes. We explain how a node d allocates data cells of its child nodes f, g, and h when d's cell is given by its parent b as $S_d = (l_d = 1, c_d = 0)$. Assume that L_{SF} is 6 and the number of available channels is 4. At the beginning of the cell allocation algorithm, d sets a list $\mathcal{A}_d =$



(a) Network topology (b) Example cell allocation

Figure 3: A small tree with 11 nodes and an example cell allocation are presented. With respect to the link db (black line), the blue colored edges denote collision links and the red lines represent 2-hop interference links. Given the cell schedule, (b) illustrates how they are actually allocated in terms of the time and frequency domains.

[2, 3, 5] since $l_b = 4$ and $l_d = 1$ are excluded, which are already allocated to the nodes b and d. Then, d allocates slots for f, g, and h as 2, 3, and 5, respectively, from the list $\mathcal{A}_d$. After this, $\mathcal{B}_v$ becomes [2, 3, 5] so as to allocate slots again when a new child node appears or the tree changes. After the slot allocation, the channel of those nodes f, g, and h is set to 1 since $1(= l_d) \bmod 4 = 1$. The black line db in Fig. 3(a) shows that by doing so, the blue colored edges which are collision links are allocated in different slots and the red lines which are interference links use different channels, so that the collision is avoided in the time domain and the interference is mitigated by using the different channels.


(i) Separate cell for downlink
(ii) Using control cell for downlink
p
v
i
j
CH
Ctl
v→p
v→i, v→j
Slot #

(a) A simple subtree (b) Downlink cell schedule

Figure 4: A simple subtree example that shows how *downlink scheduling* works. In (b), the left schedule allocates $v \to i$ and $v \to j$ at separate down link cell to support downlink traffic from v and the right schedule uses the control slot for downlink.

Scheduling for downlink traffic. We now present how our scheduling supports downlink traffic that includes query, update, and maintenance messages. We propose two methods: (i) separately allocate cell for downlink or (ii) use the control slot. As shown in Figs. 4, when downlink traffic needs to be supported reliably, we allocate downlink edges (e.g., vi and vj) at the cell where vp is already scheduled. Since our design of ALOS cell scheduling prevents child and parent nodes from transmitting simultaneously in Algorithm 1 (Line 7), at each node v, the collision between the uplink from the child nodes and the downlink from the parent node does not occur. In addition, we can also optimize the energy usage of downlink communication by controlling the access period for the downlink too. In case of low offered load over downlink, we use the control slot for downlink traffic rather than allocating additional slots so as to save more energy. Hence, SSAP can support downlink cell schedule without interrupting uplink cell schedules.

SSAP with other ALOS cell scheduling. Due to our decomposition into weakly-coupled cell scheduling and access period control, SSAP can function as a generalized framework that can include other scheduling algorithm as an ALOS scheduling module. Examples include Orchestra [8] or ALICE [18], which has focused more on autonomous operation by scheduling cells automatically based on each node or each link's ID. The key difference of our scheduling from ALICE and Orchestra lies in different reliability, which one connects to access period control for multiple QoSes.

4 DISTRIBUTED CONTROL OF ACCESS PERIOD

In this section, we formulate an optimization problem that controls access period in a distributed fashion in order to maximize the network lifetime while satisfying given latency requirements.

4.1 Formulation: α-Lifetime Maximization

Model. As we consider IoT applications that collect sensory data from sensor nodes to a base station (BS), we assume that a network $T(V, E)$ is built by some routing protocol (e.g., RPL) where V is a set of the nodes in the network and E is a set of TX-RX edges. We define a few notations in terms of a node $v \in V$ such as $\mathcal{P}_v$ which is a set of nodes in the routing path from v to BS, C_v is a set of children of v, and $\mathcal{T}_v$ is a set of nodes in the subtree of v from the tree $T(V, E)$.

As an IoT sensor, each node $v \in V$ generates traffic following a Poisson process with rate λ_v^{self}. In order to collect those data traffic from each node to BS, the amount of traffic that each node should serve is λ_v as follows:

$$\lambda_v = \lambda_v^{\text{self}} + \sum_{c \in C_v} \lambda_c$$

where we use the unit of λ_v as the number of packets per Slotframe.

Lifetime: Node and network. As a metric of energy efficiency, we consider the network lifetime, which is defined in various ways such that the time until the network cannot perform its own role (e.g., collecting sensory data). For example, max-min lifetime is the time until the first node runs out of its battery or p% lifetime is the time until p% of nodes in the network deplete their batteries. In this paper, we generalize this concept of the network lifetime as α-lifetime by mimicking the well-known α-fairness [22].

To this end, we first define the lifetime of a node using the power consumption rate, determined by (i) how often it performs listening, depending on access period n_{cv} (simply n_c) of the edge cv for all $c \in C_v$, and (ii) the rate of transmission to its parent node, i.e., λ_v. Then the power consumption rate is obtained by:

$$\mathrm{P}_v(\vec{\boldsymbol{n}}_{cv}) = \sum_{c \in C_v} \frac{\mathrm{P}_0}{n_c} + \mathrm{P}_0 \lambda_v$$

where the P_0 is the power consumption rate of a single transmission or listening, and $\vec{\boldsymbol{n}}_{cv} = [n_c]_{c \in C_v}$ denotes the vector of access period for the edges cv. Consequently, the expected lifetime of the node v is given by:

$$\mathcal{L}_v(\vec{\boldsymbol{n}}_{cv}) = \frac{\mathrm{B}_v}{\mathrm{P}_v(\vec{\boldsymbol{n}}_{cv})}$$

where B_v is an initial energy budget of each node.

Definition 4.1. *The α-lifetime of network is $\sum_{v \in V} U_\alpha(\mathcal{L}_v)$, where*

$$U_\alpha(\mathcal{L}_v) = \begin{cases} \log(\mathcal{L}_v) & \text{if } \alpha = 1, \\ \frac{\mathcal{L}_v^{1-\alpha}}{1-\alpha} & \text{otherwise} \end{cases} \quad (1)$$

We define the α-lifetime using each node v's lifetime $\mathcal{L}_v$. In terms of optimizing the α-lifetime, the case of $\alpha = 0$ implies each node behaves selfishly to maximize its own lifetime. On the other hand, as $\alpha \to \infty$, each node tries to maximize α-lifetime over the network, and thus we achieve max-min network lifetime, i.e., maximizing the minimum lifetime among nodes.

Delay. We assume that the QoS requirement of delay is given by an application. With respect to the node v, R_v is a delay requirement that requires the packet generated by v to arrive at BS within the requirement, which is often called end-to-end delay. This end-to-end delay consists of one-hop delays of all nodes in $\mathcal{P}_v$, which is the routing path from the node v to the BS as defined earlier. Let d_v be the expected one-hop delay from v to its next-hop node p (i.e., a parent node). As the definition of access period, v can access the scheduled cell in every n_v frame, which can be regarded as deterministic service rate. Thus, we model it as $M/D/1$ queue whose arrival process is Poisson with rate λ_v and the service time is determinimistic with rate $\frac{1}{n_v}$. Then, based on the queueing theory [3], we derive the expected one-hop delay d_v as follows:

$$d_v(n_v) = \frac{n_v}{2(1 - \lambda_v n_v)} \times \frac{1}{r_v} \quad (2)$$

where the last term $\frac{1}{r_v}$ is the abstracted reliability on the link vp which is given by the cell scheduling module. This reliability is introduced to the delay to guarantee the end-to-end delay under the possible retransmission. Then, the expected end-to-end delay from v to BS is $\sum_{k \in \mathcal{P}_v} d_k(n_k)$.

Problem: α-lifetime maximization. In order to achieve the primal goal of SSAP, we choose the objective function as the α-lifetime (Eq. (1)) with the delay constraints using the expected delay from Eq. (2). Then, the α-lifetime maximization problem **LIFE_MAX(α)** can be formulated as follows:

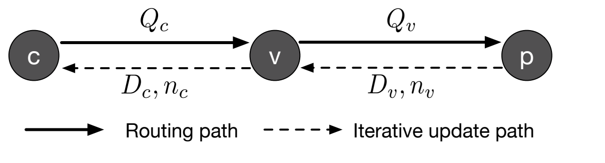
$$\textbf{LIFE\_MAX}(\alpha): \quad \max_{\vec{n}} \sum_{v \in V} U_\alpha(\mathcal{L}_v(\vec{n}_{cv})) \quad (3)$$

subject to

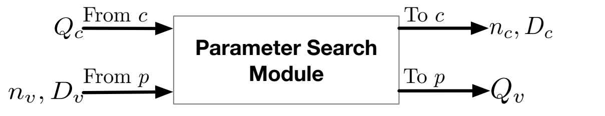
$$\sum_{k \in \mathcal{P}_v} d_k(n_k) < R_v \quad \forall v \in V \quad (4)$$

where we note that both Eq. (3) and Eq. (4) are the functions of $\vec{n}$ which is the vector of all access period over the network.

The above optimization requires to collect all information, such as delay requirements and traffic rates from all nodes in the network, to find the optimal $\vec{n}^*$ and to distribute the obtained $\vec{n}^*$ to all nodes. This centralized algorithm is not scalable nor efficient especially in low power networks for IoT applications, since the huge amount of information should be collected and distributed. Thus, we develop a distributed algorithm to solve this optimization, which is described in the following section.



(a) Distributed parameter update flow



(b) Update with local communication in terms of node v

Figure 5: (a) shows the distributed parameter update flow in terms of interaction between a node v, its parent p, and its child c. (b) describes how the parameter search algorithm updates via local communication.

4.2 Distributed Update: Rationale and Description

In this section, we propose an algorithm that is designed to solve the **LIFE_MAX(α)** problem in a distributed manner. In order to develop a distributed algorithm, we first introduce the primal-dual formulation, so that the **LIFE_MAX(α)** problem is translated into the dual problem which can be decomposed into independent sub-problems. Then we propose the parameter search algorithm that finds the optimal access period by solving each sub-problem in a distributed manner.

Dual problem for distributed optimization. To decompose the problem, we construct the dual problem from **LIFE_MAX(α)**. Let $L(\vec{n}, \vec{q})$ be the Lagrangian function with Lagrange multipliers $\vec{q}$:

$$L(\vec{n}, \vec{q}) = \sum_{v \in V} U_\alpha(\mathcal{L}_v(\vec{n}_{cv})) + \sum_{v \in V} q_v \left(R_v - \sum_{k \in \mathcal{P}_v} d_k(n_k) \right) \quad (5)$$

The Lagrangian function (Eq. (5)) can be decomposed into independent sub-functions for each node v where the control variables (i.e., $\vec{n}_{cv}$) of each sub-function are disjoint over nodes. This decomposition is possible due to the tree-shaped topology of the network (e.g., TSCH networks) where traffic direction is restricted to follow the tree structure. We can rewrite Eq. (5) as follows.

$$L(\vec{n}, \vec{q}) = \sum_{v \in V} q_v R_v + \sum_{v \in V} L_v(\vec{n}_{cv}, \vec{q})$$
$$L_v(\vec{n}_{cv}, \vec{q}) = U_\alpha(\mathcal{L}_v(\vec{n}_{cv})) - \sum_{c \in C_v} d_c(n_c) \sum_{k \in \mathcal{T}_c} q_k$$

Each node v's lifetime $\mathcal{L}_v(\vec{n}_{cv})$ only depends on its child node $c \in C_v$, and thus the Lagrangian function can be decomposed with respect to $\vec{n}_{cv}$.

On the other hand, the corresponding dual function also can be decomposed. Given $\vec{q}$, we define $\vec{n}^*(\vec{q})$ as the optimal $\vec{n}$ that maximizes $L(\vec{n}, \vec{q})$. It can be obtained by:

$$\vec{n}^*(\vec{q}) = \operatorname*{argmax}_{\vec{n}} L(\vec{n}, \vec{q})$$

Then, given the optimal $\vec{n}^*$, the dual objective function $g(\vec{q})$ can be formulated as follows:

$$g(\vec{q}) = L(\vec{n}^*(q), \vec{q}) \quad (6)$$

Algorithm 2: Parameter search algorithm of node v

```
1  while true
2    if $v$ communicates with $c$: update $\vec{n}_{cv}$
3      Receive $Q_c$ from $c$
4      $\vec{n}_{cv} \leftarrow \text{argmax}_{\boldsymbol{n}} \left[U_\alpha(\mathcal{L}_v(\vec{n})) - \sum_{c \in C_v} d_c Q_c\right]$
5      $D_c \leftarrow D_v + d_c$
6      Transmit $n_c$ and $D_c$ to $c$
7    if $v$ communicates with $p$: receive $n_v$ and update $q_v$
8      Receive $D_v$ and $n_v$ from $p$
9      $q_v \leftarrow \left[q_v - \frac{1}{t}(R_v - D_v)\right]^+$
10     $Q_v \leftarrow q_v + \sum_{c \in C_v} Q_c$
11     Transmit $Q_v$ to $p$
12 done
```

$$= \sum_{v \in V} q_v \left(R_v - \sum_{k \in \mathcal{P}_v} d_k(n_k^*) \right) + \sum_{v \in V} U_\alpha(\mathcal{L}_v(\vec{n}^*_{cv}))$$

and thus, the dual problem of **LIFE_MAX(α)** becomes:

$$\textbf{DP}: \min_{\vec{q}} g(\vec{q}) \tag{7}$$
$$\text{subject to } \vec{q} \geq 0$$

Similar to the Lagrangian function, the dual function Eq. (6) also can be decomposed into independent sub-function $g_v(q_v)$:

$$g_v(q_v) = q_v(R_v - \sum_{k \in \mathcal{P}_v} d_k(n_k))$$
$$g(\vec{q}) = \sum_{v \in V} g_v(q_v) + \sum_{v \in V} U_\alpha(\mathcal{L}_v(\vec{n}^*_{cv}))$$

By this decomposition, the function parameters $\vec{n}_{cv}$ and q_v of $L_v(\vec{n}_{cv}, \vec{q})$ and $g_v(q_v)$ are fully controlled by the single node v. This enables us to develop the distributed parameter search algorithm which solves the decomposed primal-dual problems per each node. The proposed algorithm separately maximizes $L(\vec{n}, \vec{q})$ and minimizes $g(\vec{q})$, and each node v achieves the global optimal of **LIFE_MAX(α)**, as well as that of the dual problem (Eq. (7)). We will show that this algorithm can find the global optimal parameter of the primal problem in Section 4.3.

Distributed update algorithm. Based on this dual formulation and decomposition, we propose the parameter search algorithm which runs on each node in a distributed manner. Algorithm 2 describes how this algorithm finds the optimal access period iteratively in terms of the node v.

As illustrated in Figs. 5, the parameter search module requires local communication with its parent p and its child nodes c so as to receive its access period from p and to output access period of $c \in C_v$. In order to solve the primal and dual problems, each node $v \in V$ needs to update q_v and $\vec{n}_{cv}$ iteratively.

Update $\vec{n}_{cv}$. For child nodes $c \in C_v$ to be updated, v first needs to receive Q_c (Line 3) which is the summation of delay prices q_i in $\mathcal{T}_v$, we recall that $\mathcal{T}_v$ is a set of nodes in v's subtree. Then, for the given Q_c, v analytically computes $\vec{n}_{cv}$ that maximizes the Lagrangian function (i.e., equivalent to maximize the α-lifetime) (Line 4). We note that this update procedure is the most computationally expensive (e.g., Newton's method), and thus it should be verified that this update can be computed on a low power IoT device (See Section 5.3). As receiving D_v (i.e., the expected end-to-end delay from v to BS) from its parent p, it can compute D_c by adding D_v to d_c (Line 5). Then, v distributes $n_c \in \vec{n}_{cv}$ and D_c to c (Line 6).

Receive n_v and update q_v. In terms of v and its parent p, v receives the updated n_v by the above update iteration done by p, and concurrently, the end-to-end delay D_v is also given by p (Line 8). Then the delay price q_v can be obtained by the gradient descent method with a step size $\frac{1}{t}$ using D_v (Line 9). This delay price represents how much the delay constraint is violated, and thus by doing so, the dual function is also minimized. Lastly, v needs to calculate Q_v and transmit it to p (Line 10-11) so as to let p update v's access period. Even though Q_v is the accumulated summation over all nodes in its subtree $\mathcal{T}_v$, we compute Q_v efficiently by using the tree structure; thus v computes Q_v as the summation of q_v and received Q_c from its child nodes, which are accumulated recursively (Line 10).

4.3 Convergence and Implementation Issues

In this section, we present an analytical result about the convergence of our distributed update of access period in Algorithm 2, and it is followed by the discussion of implementation issues.

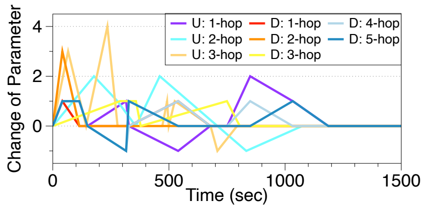
THEOREM 4.1. *The update algorithm of access period in Algorithm 2 converges to the solution of α-lifetime maximization problem **LIFE_MAX(α)**.*

The proof is presented in Appendix.

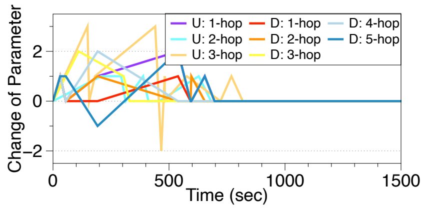
Implementation issues. In order to implement SSAP efficiently on IoT devices, we propose the following implementation details.

Communication for local information exchange. In order to run this algorithm, each node requires two directional communication: one is to its parent and the other is to its child. As the TSCH and RPL standard indicate, there exist various types of control packets (e.g., EB for TSCH, DIO and DAO for RPL, respectively). Hence, those local information can be exchanged through information embedding on those control packets. Another approach is using piggybacking. We exploit the unicast and ACK operation (i.e., a data packet followed by an ACK) which should be performed in the TSCH network. According to the standard, each unicast communication must have Acknowledgement mechanism to maintain the global synchronization. Thus, in terms of a node v, we embed Q_v to the packet header and receive D_v and n_v which are embedded in the ACK header. Although it obviously induces more overhead on each communication, we also show that this algorithm converges in quite short time; thus, this overhead is negligible compared to the improvement by running this algorithm, as described in later sections.

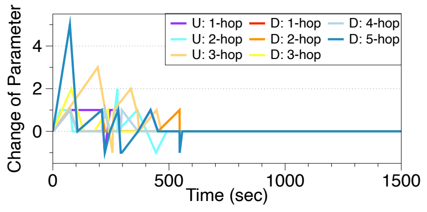
Algorithm complexity. In order to validate low overhead of SSAP, we evaluate the convergence of the parameter search algorithm using simulations (the detailed simulation environments are described in Section 5). In this evaluation, we use the above embedding mechanism on data packets and ACKs. As shown in Figs. 6, in a simulation topology, **SIM** with 43 nodes, we measure the change of access period on each node with varying Poisson rate λ where each node on different hop counts is illustrated for simplicity (U and D denote the location of each node in the upper and down side of the topology).



(a) Poisson rate $\lambda = 0.667$



(b) Poisson rate $\lambda = 1$



(c) Poisson rate $\lambda = 2$

Figure 6: Evaluation on the convergence of the parameter search algorithm.

As illustrated, under all λ from 0.667 to 2 packets/min, this algorithm converges within 1200 seconds. More precisely, in the case of $\lambda = 2$, since they communicate 2 times per minute on average, it only requires less than 20 information exchanges to be converged. Thus, it indicates small overhead of SSAP and robustness to network changes due to the fast convergence. In addition to the convergence, for each of information exchange, we analyze the overhead on headers of a packet and an ACK. In each iterative update, a node v only needs to transmit and receive three variables (i.e., n_v, D_v, and Q_v). Since we fully utilize the tree structure to efficiently collect those information, the embedded overhead can be minimized.

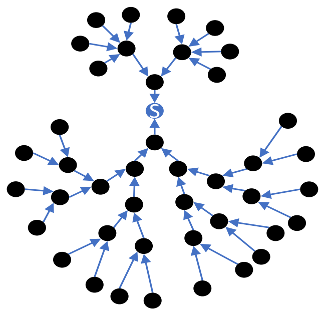
Adapt to environmental changes. Although SSAP is designed to support static applications without mobility, it is also crucial to adapt to environmental changes, e.g., topology or delay requirement changes. Any node v in SSAP has an ability to adapt to any change on the paths that the node belongs to if this change incurs modified delay D_v or delay price Q_v, otherwise v do not need to update. We note that this local change only requires to update cell scheduling and parameter search with respect to the affected nodes due to the distributed operation of each protocol. To validate adaptivity, we perform two evaluations: (i) Under the same simulation environment as the above, we make topology change of one node[4]; then, it requires the convergence time of 195 seconds on average, while the other nodes which are not affected by the change do not require update. (ii) When the delay requirement of all nodes changes from 600 ms to 1200 ms, it takes 750 seconds on average to converge again. Accordingly, we show that SSAP quickly adapts to small changes and is designed to respond the environmental big change as it updates and converges over again.

4.4 Putting Everything Together

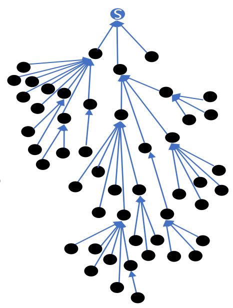
We now present how SSAP works by putting the cell scheduling and the parameter search algorithms together.

After deploying a TSCH network (e.g., a base station and sensor nodes), the base staion of the network starts to initiate network advertisement by transmitting EB and RPL packets. As some nodes join the network following TSCH and RPL standards, they start to run the cell scheduling algorithm based on their routing tree which is built by RPL. From the base station, each node v schedules its child nodes $c \in C_v$, so that each link cv can communicate at slot l_c and channel c_c according to $S_c = (l_c, c_c)$. After each node is scheduled, it also runs the parameter search algorithm. Since this algorithm runs in a distributed manner, each node does not have to wait until the tree is stabilized but is able to start updates only if it is scheduled. Thus,

[4]A node apart from the BS by 3 hops switches its parent to the node in the same level.



(a) **SIM:** 43 nodes



(b) **EXP:** 50 nodes

Figure 7: Evaluation topologies: (a) SIM for simulations and (b) EXP for experiments where S denotes a base station.

in terms of v, the parameter search algorithm iteratively updates access period for all child nodes c connected to v. Assume that the access period n_c of c is given by this algorithm. Then, when the link cv is scheduled at S_c, this schedule is activated only in every n_c Slotframe. It means that the communication on the link cv can occur only when the cell is activated, otherwise the nodes c and v save energy by doing nothing. More precisely, the schedule of the link cv is activated if SFN mod n_c is equal to 0, otherwise the schedule is inactivated where SFN is obtained by ASN[5] divided by L_{SF}. As a result, with ALOS cell scheduling, each link has enough chances for communication so as to guarantee its delay requirement, while the lifetime of the network can be improved by the heterogeneous access period.

5 IMPLEMENTATION AND EVALUATION

5.1 Implementation

We implement our scheduling algorithm SSAP on Contiki OS [7], which is an open source operating system for IoT devices. Based on the implementation of TSCH in the Contiki OS, we modify and implement our designs of SSAP. To construct the network topology, the objective function of RPL is chosen as MRHOF [11] which builds a tree using hop counts and ETX. To exchange the update information for SSAP, we use piggybacking mechanism to data packets and ACKs. We evaluate SSAP using Cooja simulator in the Contiki OS to test various scenarios under a controlled environment and use a real testbed in IoT-LAB [4] with 50 M3 nodes.

[5]As defined in the TSCH standard, ASN (Absolute Slot Number) is a global slot number which increases by 1 for every elapsed slot.

5.2 Evaluation on Simulation

Simulation environment. As a simulation environment, in order to emulate mission-critical applications, we consider application scenarios where data packets are generated Poisson randomly with arrival rate λ and delay requirements are given by the application. The hyperparameter of α-lifetime is chosen by 5, which is empirically chosen for the performances. Our simulation topology **SIM** is shown in Fig. 7(a) which is composed of 43 nodes in an unbalanced tree. We choose the unbalanced tree to emphasize heterogeneity requirement among nodes. Main performance metrics of our evaluation are composed of (i) the maximum end-to-end delay from a source to a base station (BS), (ii) the normalized lifetime which reflects the time until some node runs out of its battery at first, calculated by the ratio between the intial energy budget and the energy consumption by the radio, (iii) the average duty cycle (i.e., percentage of time the radio is on), and (iv) the Packet Reception Ratio (PRR) which denotes the the ratio between the number of received packets (at BS) and the number of generated packets (at a source). We compare the performance of SSAP with the state-of-the-art distributed scheduling protocols, Orchestra [8] with sender-based scheduling (Orche-SB) and ALICE [18].

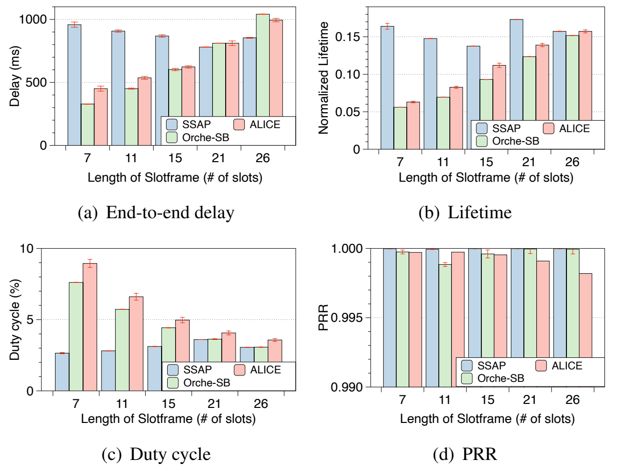


Figure 8: Evaluation on the impact of the length of Slotframe. Red error bars show standard deviation.

(a) End-to-end delay (b) Lifetime (c) Duty cycle (d) PRR

Impact of Slotframe length. In this evaluation, we show the impact of the length of Slotframe by varying the length from 7 to 26. Figs. 8 demonstrate the main four metrics of SSAP compared to those of Orhce-SBS and ALICE where the arrival rate $\lambda = 2$ packets/min and the delay requirement D is given by 1000 ms. Since SSAP has an ability to adapt the parameters over the network to guarantee the delay requirement, regardless of the length of Slotframe, Fig. 8(a) shows that SSAP achieves the end-to-end delay under 1000 ms. We note that the measured end-to-end delay is the maximum over the nodes in the network, so that we claim that the application delay requirement is guaranteed. While guaranteeing the delay requirement, SSAP also maximizes the lifetime and minimizes the energy consumption, represented by the normalized lifetime and the duty cycle in Figs. 8(b) and 8(c), respectively. Compared to SSAP, the performances of Orche-SB and ALICE depend highly on the length of Slotframe since they always allocate a slot in each Slotframe regardless of their heterogeneity. For some Slotframe length (e.g., 7 or 11), the delay performance is small enough to guarantee the delay requirement; however, they consume a lot of energy so that the lifetime is too short compared to that of SSAP. On the other hand, the large Slotframe length (e.g., 26) can improve the lifetime, but the delay requirement cannot be guaranteed. Fig. 8(d) shows the PRR performance where SSAP shows the highest PRR due to its cell allocation algorithm compared to Orche-SB and ALICE which may have slot conflict.

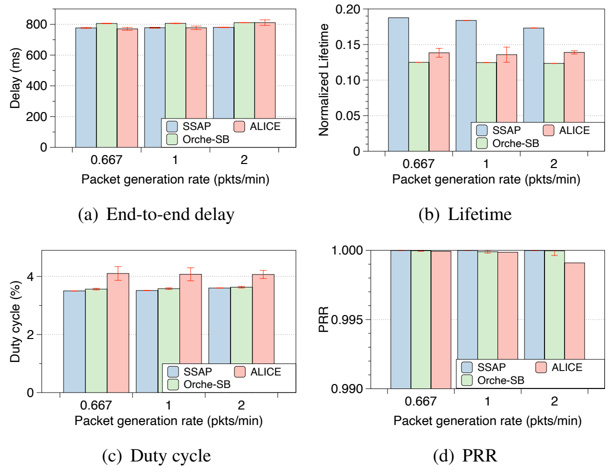
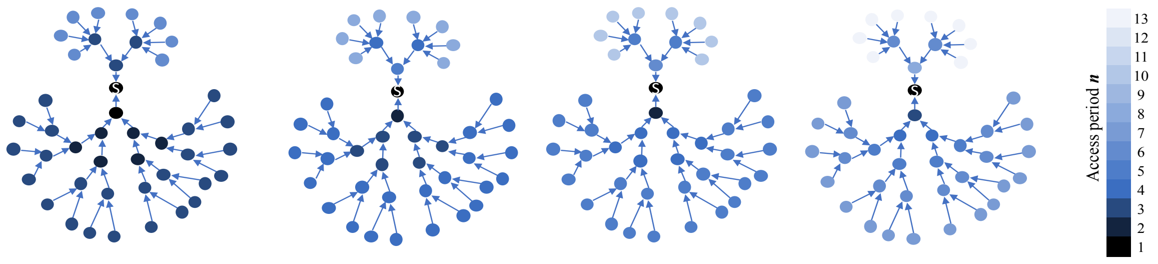


Figure 9: Evaluation on the impact of traffic demand. Red error bars show standard deviation.

(a) End-to-end delay (b) Lifetime (c) Duty cycle (d) PRR

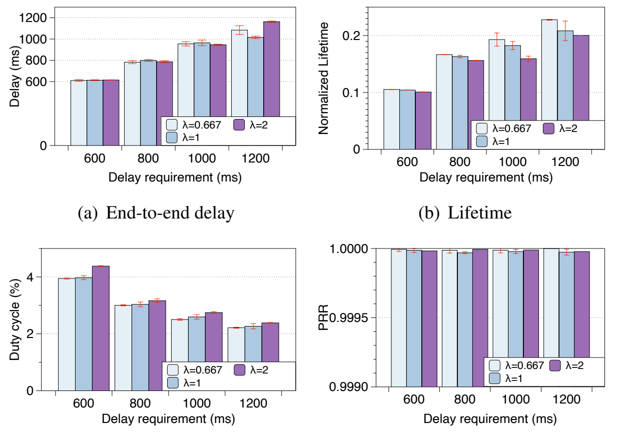
Impact of traffic demand. To evaluate the impact of different application traffic, we measure the main performance metrics by varying the arrival rate $\lambda = 0.667, 1$, and 2 packets/min. The delay requirement is given by 1000 ms and the length of Slotframe for Orche-SB and ALICE is chosen by $L_{\mathrm{SF}} = 21$ which achieves the delay requirement while maximizing the lifetime. SSAP also chooses the same Slotframe length for fair comparison. As shown in Fig. 9(a), all protocols guarantee the end-to-end delay requirements. However, they clearly show different performances on the lifetime and duty cycle. In particular, SSAP achieves the much longer lifetime by exploiting heterogeneous access period over all nodes, while the other protocols cannot maximize the lifetime due to the same opportunity of communication for the all nodes. This is demonstrated in Fig. 9(c) which shows that SSAP and the other protocols have almost the same average duty cycle (ALICE is a little higher due to its link-based scheduling). By varying the arrival rates, SSAP still achieves the better performances compared to those of Orche-SB and ALICE, which means SSAP can support various IoT application traffic patterns.

Impact of delay requirement. Since the design goal of SSAP is to guarantee the delay requirement while maximizing the lifetime, we show that SSAP adapt to the different delay requirements. Figs. 11 demonstrate the evaluation result of SSAP with various delay requirements 600, 800, 1000, and 1200 ms where the length of Slotframe is 7 and the arrival rate is chosen from 0.667, 1, and 2 packets/min. As shown in Fig. 11(a), for each of the delay requirement, SSAP properly adapts to that requirement by searching the



(a) R = 600 ms (b) R = 800 ms (c) R = 1000 ms (d) R = 1200 ms

Figure 10: Evaluation on heterogeneous access period by varying delay requirements.



(a) End-to-end delay (b) Lifetime

(c) Duty cycle (d) PRR

Figure 11: Evaluation on the impact of varying delay requirements. Red error bars show standard deviation.

optimal access period in a distributed manner. We also illustrate the heterogeneous access period of each node in Fig. 10. While the end-to-end delay requirement is tightly achieved, as the requirement becomes loose, SSAP improves the network lifetime and the energy-efficiency significantly as shown in Figs. 11(b) and 11(c), while the PRR performance is high enough to support the mission-critical applications (Fig. 11(d)).

Heterogeneous access period. With the same setting in the above evaluation of varying delay requirements, we visualize how access period is determined adaptively in different delay requirements. Figs. 10 illustrate heterogeneous access period on each node with the delay requirements from 600 to 1200 ms where $\lambda = 1$ packets/min and the heterogeneous parameters are represented by different colors as shown in the right-hand side of Fig. 10. As the delay requirement becomes loose, SSAP finds the larger access period for each node to maximize the network lifetime. Comparing the nodes from the small subtree to the other nodes, they also have larger access period since it is much easy to guarantee the delay requirement due to the short path so as to maximize the lifetime. By combining both result from the evaluation of varying the requirement and the heterogeneous access period, it clearly is shown that SSAP improves the network lifetime by adaptively finding the best access period over the network.

5.3 Evaluation on Testbed

To evaluate SSAP in a real testbed, we implement SSAP on the M3 nodes. As explained earlier, we choose IoT-LAB as the real testbed and evaluate SSAP with Orche-SB and ALICE. We build the testbed with the 50 M3 nodes and the RPL protocol with MRHOF (i.e., the objective function that uses ETX and hop count as routing metrics) is used to build the routing tree as shown in Fig. 7(b) where we set TX power of each node to -17 dBm to build a multi-hop network. Before evaluating the main performance metrics, we first evaluate the feasibility of SSAP on real IoT devices. Since the parameter search module requires optimization computation to update access period over its child nodes, this computation should be done at the IoT device within some reasonable time.

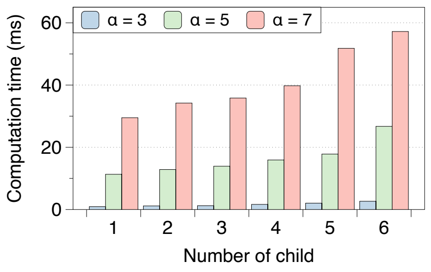
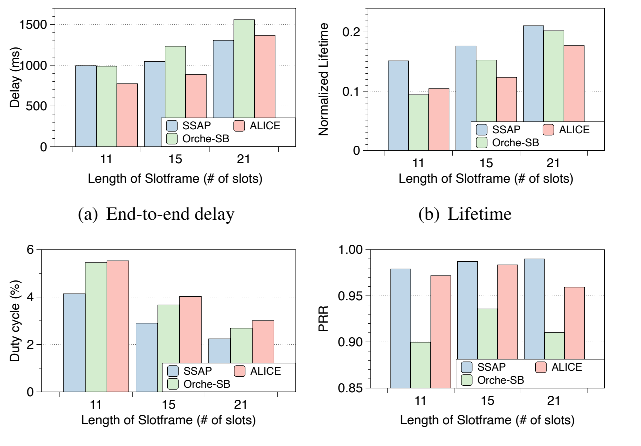


Figure 12: Computation time on M3 nodes from IoT-LAB. Standard deviation is omitted due to its negligible variance.

Computation complexity. As proposed in previous sections, the parameter search algorithm of SSAP needs to update access period using an analytic method in a distributed manner. To show the feasibility of SSAP in low power IoT networks, we validate the computation complexity of our update algorithm on the low power M3 device with 32 MHz MCU. Fig. 12 shows the computation time of a node to obtain the access period for all its child nodes. As the number of child nodes increases, the computation time increases for all α. However, they are smaller than 60 ms and for our choice of $\alpha = 5$, the computation time is short enough to be computed on real devices. Furthermore, our previous evaluation shows that about 20 iterations are enough to converge access period. Thus, we claim that this computation overhead is negligible compared to the improvement of the network lifetime through SSAP.

Performance evaluation on the real testbed. Figs. 13 demonstrate the main performance metrics as the same as those of the simulations comparing SSAP to the other protocols. With varying the length of Slotframe which impacts highly on all of the performances (from 11 to 21), we let each node generates data packets with Poisson rate $\lambda = 1$ packets/min. We assume that the delay requirement is given by 1200 ms for all nodes. As shown in Fig. 13(a), SSAP achieves the delay requirement since the maximum end-to-end delay is small than 1200 ms except the case of Slotframe length is 21.



(a) End-to-end delay (b) Lifetime

(c) Duty cycle (d) PRR

Figure 13: Evaluation on the real testbed with varying the length of Slotframe.

This is because considering the result of Orche-SB and ALICE, it is impossible to achieve this requirement due to the topology when the Slotframe length is too long (i.e., 21). In terms of the lifetime and energy efficiency (i.e., radio duty cycle), similar to the simulation results SSAP achieves the longest network lifetime and the minimum average duty cycle compared to the others where the normalized lifetime in Fig. 13(b) is obtained by 1 divided by the value of the node's duty cycle which has the largest value among nodes. Fig. 13(d) shows the PRR performances of those protocols. In the real testbed, not only the topology is more dense but also link loss and interference are much severe compared to those of the simulation, and thus they suffer from a lot of packet drops. However, SSAP shows that the achieved reliability is still better than that of the others due to its conflict-free scheduling. As a result, we claim that under deployed environments SSAP maximizes the network lifetime while it achieves the delay requirements; thus, SSAP can support a lot of mission-critical IoT applications with minimizing network maintenance cost.

6 CONCLUSION

This paper proposed ***SSAP*** which is a novel scheduling algorithm for TSCH. SSAP achieved the QoS of expected latency while maximizing energy efficiency, quantified by the α-lifetime. In order to develop a distributed protocol, we first introduced the tunable parameter which trade offs the energy efficiency and delay by controlling the period of schedule activation, called access period. Using access period, we decomposed the original scheduling problem into ALOS cell scheduling and the parameter search, so that given QoS requirement, SSAP can find the best schedule and access period in a distributed manner.

In order to evaluate SSAP, we implemented SSAP in the Contiki OS and performed extensive simulations and experiments on the performance of the lifetime, end-to-end delay, and reliability (i.e., PRR). Compared to the state-of-the-art protocols, we showed that SSAP satisfies the QoS requirement of delay while achieving at most 1.6 times longer lifetime, while the others cannot guarantee the QoS or consume too much energy. Moreover, the complexity evaluation and adaptivity to environmental changes of SSAP demonstrated the practicality of SSAP when it is deployed in situ.

ACKNOWLEDGMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2016-0-00160,Versatile Network System Architecture for Multi-dimensional Diversity) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00772,Development of an ultra-Low power(500uW) long range radio for ultra-small IoT based on new Two-Tone OOK modulation scheme). We would like to thank to the anonymous reviewers and shepherd for the valuable comments.

A APPENDIX: PROOF OF THEOREM 4.1

Theorem 1 from [21] states that the distributed gradient descent algorithm described in Algorithm 2 will converge to the globally optimal point with sufficiently small step size, when the following conditions holds for all node:

(1) The primal parameter x is bounded away from zero by positive constants m and $M : m \leq x \leq M$ where $m > 0$.
(2) The gradient of primal parameter x with respect to the dual parameter q exists and is bounded between zero and a negative constant $\mathrm{C}_1 : \mathrm{C}_1 < \frac{\partial x}{\partial q} \leq 0, \forall q$.
(3) The utility function is twice continuously differentiable, strictly concave and increasing in the range of the primal parameter.

We now adopt the above result and show that our algorithm also converges to the optimal point by proving that the above conditions are satisfied. Given a tree network $T(V, E)$, for an edge $vp \in E$, let n_{vp} (simply n_v) be the primal parameter (access period), and q_v be the dual parameter (delay price) of a node $v \in V$. We now show the conditions are satisfied for every node $v \in V$:

(1) The primal parameter is at least 1 since each node is assigned at most 1 slot per Slotframe. The primal parameter is also less than R_v which is the delay constraint. Thus, $1 \leq n_v \leq R_v$.
(2) While traditional NUM uses linear sum of primal parameters as constraints, our problem considers the sum of non-linear delay functions as a constraint. Hence, the second condition is rewritten using the delay function : $\mathrm{C}_1 < \frac{\partial d_v(n_v)}{\partial q_k} \leq 0$, $\forall k \in V$.
It is easy to see that n_v and q_k are independent if k is not in the subtree $\mathcal{T}_v$ of v, so that the gradient $\frac{\partial d_v(n_v)}{\partial q_k}$ is 0 for all such k. Hence, we only consider q_k for $k \in \mathcal{T}_v$, which is a dual parameter of nodes $k \in \mathcal{T}_v$. The gradient of the delay $d_v(n_v)$ with respect to q_k can be expressed as follows:

$$\frac{\partial d_v}{\partial q_k} = \frac{1}{4(1-\lambda_v n_v)((1-\lambda_v n_v)^3 \frac{\partial^2 U_\alpha(\mathcal{L}_p)}{\partial^2 n_v} - \lambda_v Q_v)}$$

Note that $U_\alpha(\mathcal{L}_p)$ is used since the n_v affects the parent's lifetime only, and the equation does not depends on the choice of k. From the basic assumption, the delay of each node should be less than the delay constraint. Thus, we have the following bounds:

$$d_v(n_v) = \frac{n_v}{2(1-\lambda_v n_v)} < R_v \text{ and } 1 - \lambda_v n_v > \frac{1}{2R_v}$$

The second-order derivative of the objective function of the parent node $U_\alpha(\mathcal{L}_p)$ with respect to n_v can be expressed as follows:

$$\frac{\partial^2 U_\alpha(\mathcal{L}_p)}{\partial^2 n_v} = -\left(\frac{1}{\sum_{c\in C_p}\frac{1}{n_c}+\lambda_v}\right)^{-\alpha+3}\cdot\left(\frac{1}{n_v^3}\right) \cdot\left(\frac{\alpha}{n_v}+2\lambda_v+\sum_{c\in C_p, c\neq v}\frac{2}{n_c}\right)$$

Since each n_v is bounded between strictly positive bounds: $1 \le n_v \le R_v$, it is easy to see that the second-order derivative of the $U_\alpha(\mathcal{L}_p)$ is also bounded between some strictly negative constants C_2 and C_3: $C_2 < \frac{\partial^2 U_\alpha(\mathcal{L}_p)}{\partial^2 n_v} < C_3 < 0$. Then, we have the following bound:

$$\frac{4R_v^4}{C_3} < \frac{\partial d_v}{\partial q_k} \le 0$$

(3) It is easy to see that the utility function is increasing and twice continuously differentiable from Eq. (1).
For the strict concavity, it is sufficient to prove that each $U_\alpha(\mathcal{L}_v)$ is strictly concave, then the objective function as a summation of strictly concave functions is also strictly concave. For each v, let $A_v = \sum_{c\in C_v}\frac{1}{n_c}+\lambda_v$, then we have the following hessian matrix for each $U_\alpha(\mathcal{L}_v)$:

$$\mathbf{H}_v = -\frac{(B/k)^{-\alpha+1}}{(A_v)^{-\alpha+3}}\left(\alpha g^T g + 2\lambda_v Diag(\frac{1}{n^3}) + \sum_{i\in C_v}\sum_{j\neq i} h_{ij}^T h_{ij}\right)$$

where $h_{ij} = \left[0...\frac{1}{n_i}...0...\frac{-1}{n_j}...0\right]$, g is a vector whose i th entry is $\frac{1}{n_i^2}$, and Diag$(\frac{1}{n^3})$ is a diagonal matrix whose (i,i) th entry is $\frac{1}{n_i^3}$. Diagonal matrix and each $x^T x$ with $x \neq 0$ is positive definite, so that the hessian matrix is negative definite as a negative summation of positive definite matrices, and therefore $U_\alpha(\mathcal{L}_v)$ is strictly concave.

REFERENCES

[1] 802.15.4e Task Group. IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. *IEEE Std 802.15.4e*, 2012.
[2] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler. Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things. In *Proc. of IEEE WoWMoM*, 2013.
[3] I. Adan and J. Resing. *Queueing theory*. Eindhoven University of Technology Eindhoven, 2002.
[4] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, et al. FIT IoT-LAB: A large scale open experimental IoT testbed. In *Proc. of IEEE WF-IoT*, 2015.
[5] G. Daneels, B. Spinnewyn, S. Latré, and J. Famaey. ReSF: Recurrent Low-Latency Scheduling in IEEE 802.15. 4e TSCH networks. *Ad Hoc Networks*, 69:100–114, 2018.
[6] M. Domingo-Prieto, T. Chang, X. Vilajosana, and T. Watteyne. Distributed pid-based scheduling for 6tisch networks. *IEEE Communications Letters*, 20(5):1006–1009, 2016.
[7] A. Dunkels, B. Grönvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Proc. of IEEE LCN*, 2004.
[8] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *Proc. of ACM SenSys*, 2015.
[9] S. C. Ergen and P. Varaiya. TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks*, 16(4):985–997, 2010.
[10] X. Fafoutis, A. Elsts, G. Oikonomou, R. Piechocki, and I. Craddock. Adaptive static scheduling in IEEE 802.15. 4 TSCH networks. In *Proc. of IEEE WF-IoT*, 2018.
[11] O. Gnawali. The minimum rank with hysteresis objective function. *IETF RFC6719*, 2012.
[12] R. T. Hermeto, A. Gallais, and F. Theoleyre. Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey. *Computer Communications*, 2017.
[13] I. Hosni and F. Théoleyre. Self-healing distributed scheduling for end-to-end delay optimization in multihop wireless networks with 6TiSCh. *Computer Communications*, 110:103–119, 2017.
[14] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi. Fast data collection in tree-based wireless sensor networks. *IEEE Transactions on Mobile Computing*, 11(1):86–99, 2012.
[15] ISA. Wireless system for industrial automation: process control and related applications. ANSI/ISA-100.11a-2011.
[16] S. Jeong, J. Paek, H.-S. Kim, and S. Bahk. Tesla: Traffic-aware elastic slotframe adjustment in tsch networks. *IEEE Access*, 7:130468–130483, 2019.
[17] J. Jung, D. Kim, J. Hong, J. Kang, and Y. Yi. Parameterized Slot Scheduling for Adaptive and Autonomous TSCH Networks. In *Proc. of IEEE INFOCOM Workshop on MiSeNet*, 2018.
[18] S. Kim, H.-S. Kim, and C. Kim. ALICE: autonomous link-based cell scheduling for TSCH. In *Proc. of ACM/IEEE IPSN*, 2019.
[19] R.-A. Koutsiamanis, G. Z. Papadopoulos, X. Fafoutis, J. M. Del Fiore, P. Thubert, and N. Montavont. From best effort to deterministic packet delivery for wireless industrial iot networks. *IEEE Transactions on Industrial Informatics*, 2018.
[20] LAN/MAN Standards Committee and others. IEEE Standard for Local and metropolitan area networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Computer Society Approved*, 2011.
[21] S. H. Low and D. E. Lapsley. Optimization flow control. i. basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.
[22] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000.
[23] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne. Label switching over IEEE802. 15.4 e networks. *Transactions on Emerging Telecommunications Technologies*, 24(5):458–475, 2013.
[24] R. T. Najafabadi, M. Nabi, A. Basten, and K. Goossens. Hybrid Timeslot Design for IEEE 802.15. 4 TSCH to Support Heterogeneous WSNs. In *Proc. of IEEE PIMRC*. IEEE, 2018.
[25] M. Nobre, I. Silva, and L. A. Guedes. Routing and scheduling algorithms for WirelessHARTNetworks: a survey. *Sensors*, 15(5):9703–9740, 2015.
[26] S. Oh, D. Hwang, K.-H. Kim, and K. Kim. Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH. *Sensors*, 18(4):1209, 2018.
[27] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia. Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. In *Proc. of IEEE PIMRC*, 2012.
[28] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. A. Grieco. Autonomous and traffic-aware scheduling for TSCH networks. *Computer Networks*, 135:201–212, 2018.
[29] L. Shi and A. O. Fapojuwo. Tdma scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(7):927–940, 2010.
[30] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. WirelessHART: Applying wireless technology in real-time industrial process control. In *Proc. of IEEE RTAS*, 2008.
[31] R. Soua, P. Minet, and E. Livolant. MODESA: an optimized multichannel slot assignment for raw data convergecast in wireless sensor networks. In *Proc. of IEEE IPCCC*, 2012.
[32] R. Soua, P. Minet, and E. Livolant. Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15.4e TSCH networks. *Transactions on Emerging Telecommunications Technologies*, 27(4):557–575, 2016.
[33] X. Wang, D. Wang, H. Zhuang, and S. D. Morgera. Fair energy-efficient resource allocation in wireless sensor networks over fading tdma channels. *IEEE Journal on Selected Areas in Communications*, 28(7):1063–1072, 2010.
[34] Y. Wu, J. A. Stankovic, T. He, and S. Lin. Realistic and efficient multi-channel communications in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2008.
[35] P. Zand, A. Dilo, and P. Havinga. D-MSR: A distributed network management scheme for real-time monitoring and process control applications in wireless industrial automation. *Sensors*, 13(7):8239–8284, 2013.
[36] Y. Zhang, C. Chen, and S. Zhu. An adaptive distributed scheduling algorithm for ieee 802.15. 4e tsch protocol. In *Proc. of IEEE ISAS*, 2019.