ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/331036806

ALICE: autonomous link-based cell scheduling for TSCH

Conference Paper · April 2019

DOI: 10.1145/3302506.3310394

CITATIONS
0

READS
334

3 authors, including:

Hyung-sin Kim
Google Inc.
54 PUBLICATIONS 446 CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Bletooth Low Energy (BLE) for IoT View project

Project MarketNet View project

All content following this page was uploaded by Hyung-sin Kim on 12 February 2019.

The user has requested enhancement of the downloaded file.

ALICE: Autonomous Link-based Cell Scheduling for TSCH

Seohyang Kim
Department of Computer Science
Seoul National University
shkim@popeye.snu.ac.kr

Hyung-Sin Kim
Computer Science Division
University of California, Berkeley
hs.kim@cs.berkeley.edu

Chongkwon Kim
Department of Computer Science
Seoul National University
ckim@snu.ac.kr

ABSTRACT

Although low-power lossy network (LLN), at its early stage, commonly used asynchronous link layer protocols for simple operation on resource-constrained nodes, development of embedded hardware and time synchronization technologies made Time-Slotted Channel Hopping (TSCH) viable in LLN (now part of IEEE 802.15.4e standard). TSCH has the potential to be a link layer solution for LLN due to its resilience to wireless interference (e.g., WiFi) and multipath fading. However, its slotted operation incurs non-trivial *cell scheduling overhead*: two nodes should wake up at a time-frequency cell *together* to exchange a packet. Efficient cell scheduling in dynamic multihop topology in wireless environments has been an open issue, preventing TSCH's wide adoption in practice. This work introduces ALICE, a novel autonomous link-based cell scheduling scheme which allocates a unique cell for each *directional link* (a pair of nodes and traffic direction) by closely interacting with the routing layer and using only local information, without any additional communication overhead. We implement ALICE on Contiki and evaluate its effectiveness on the IoT-LAB public testbed with 68 nodes. ALICE generally outperforms Orchestra (the state-of-the-art method) and even more so under heavy traffic and high node density, increasing throughput by 2 times with 98.3% reliability and reducing latency by 70%, route changes by 95%, and radio duty cycle by 35%. ALICE can serve as an autonomous scheduling framework, which paves the way for TSCH-based LLN to go on.

CCS CONCEPTS

• **Networks → Link-layer protocols**; *Cross-layer protocols*; Network performance evaluation;

KEYWORDS

Internet of Things, Low-power Lossy Network, TSCH, RPL, IPv6, Scheduling

ACM Reference Format:
Seohyang Kim, Hyung-Sin Kim, and Chongkwon Kim. 2019. ALICE: Autonomous Link-based Cell Scheduling for TSCH. In *The 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '19), April 16–18, 2019, Montreal, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3302506.3310394

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
IPSN '19, April 16–18, 2019, Montreal, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6284-9/19/04…$15.00
https://doi.org/10.1145/3302506.3310394

1 INTRODUCTION

Low power and lossy network (LLN) comprised of many resource-constrained embedded devices aims to support various applications, such as environment monitoring [6], factory automation [28], smart hospital [20], and smart market [18]. Due to the limited capability of embedded devices, the initial paradigm for LLN was that it should use simple, lightweight, and asynchronous protocols, which are significantly different from those used in conventional networks such as TCP/IP, LTE, and WiFi. However, the development of low-cost hardware and low-power network technologies has shifted this paradigm [19]. For example, IETF standardized 6LoWPAN in 2007 [46] and IPv6 Routing Protocol for LLNs (RPL) in 2012 [29], enabling resource-constrained nodes to exchange IPv6 packets in LLN and bringing Internet of Things (IoT).

Not only that, along with the latest time synchronization techniques with embedded devices [49][14][5], it has been more commonplace for LLN to exploit synchronous link layer protocols for energy efficiency and reliability. Specifically, IEEE recently standardized Time-Slotted Channel Hopping (TSCH) [51] as part of the IEEE 802.15.4e standard [50]. It provides synchronous node wake-up for low energy consumption and frequency channel hopping for reliable communication over lossy wireless links [40]. Due to its potential, TSCH has received a significant attention from both industry and academia: WirelessHART [44] and ISA100.11a [2] are designed based on TSCH to support industrial applications, and some companies, such as Analog Devices Inc., provide industrial solutions based on TSCH. It is also implemented on popular open sourced OSes, such as Contiki [4] and OpenWSN [41], which has spawned substantial research.

Challenges (Cell Scheduling). TSCH converts wireless channel into two-dimensional (time and frequency) space called *slotframe* which consists of multiple cells. Each node should properly schedule activation or deactivation of each cell to send/receive packets to/from its neighbor nodes (more details in Section 2). This cell scheduling issue is one of the subtlest aspects of TSCH, which heavily impacts reliability and latency of packet delivery and radio duty-cycle. Given that LLN has a strict resource constraint and time-varying link characteristics, a cell scheduling method should be *adaptive to dynamic multihop network topology with minimal scheduling overhead.*

A number of studies have investigated this issue, categorized into (1) centralized, (2) distributed, and (3) autonomous approaches (Section 6). The centralized approaches make the border router gather network information and schedule TSCH cells for all nodes [43, 53], which requires an end-to-end communication for each cell scheduling. These methods cannot timely adjust cell schedules to varying topology and incur communication overhead, not applicable to large-scale deployment or dynamic link environments. In the distributed approaches [3, 22, 34, 42], each node negotiates with its neighbors to determine which cells to use for communicating with each of them, incurring additional communication overhead during the negotiation procedure. As a breakthrough, Orchestra [47], an *autonomous* cell scheduling method considering RPL routing topology, was recently proposed and shown to operate on real embedded devices. However, we reveal that its *node-based scheduling* method inefficiently utilizes the two-dimensional slotframe space and incurs transmission contention, degrading performance under heavy traffic load and/or high node density (Section 3).

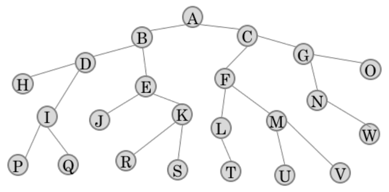
Note that traffic demand for LLN has been much more increased these days, such as large-scale deployments (e.g., CISCO's Connected Grid mesh with thousands of nodes [45]) and frequent data-gathering IoT applications (e.g., machine or structure health monitoring with acceleration and vibration [13, 35]). Emerging machine learning and artificial intelligence techniques expect to receive IoT sensor data with more frequency (frequent sensing) and density (dense node deployment) for meaningful analysis. In addition, RPL forces a small number of nodes to deliver much heavier load than others due to the load balancing problem [16, 17], increasing link layer's burden. Therefore, cell scheduling has to evolve further to provide *reliability, throughput, and low energy consumption together.*

Approach. To address the issues, we introduce ALICE, a novel autonomous link-based TSCH cell scheduling. In contrast to Orchestra (node-based scheduling), ALICE allocates a TSCH cell not for each node but for *each directional link* (i.e., a pair of nodes and traffic direction). Specifically, ALICE (1) tightly interacts with the RPL routing, (2) allows a node with more RPL neighbors (parent and children) to have more unique TSCH cells, (3) prevents upstream and downstream traffic from bothering each other, (4) utilizes multiple channels simultaneously, and (5) incurs *zero* additional overhead for cell scheduling. A novel design is provided to achieve these five features together, which utilizes routing (link ID), time (slotframe number) and traffic information (traffic direction) (Section 4).

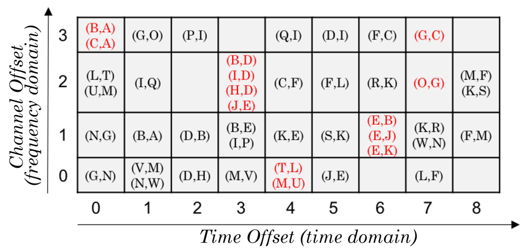
We implement ALICE on Contiki and evaluate its performance on IoT-LAB [8] (an open LLN testbed) with 68 embedded devices (Section 5). The experimental results show that ALICE reliably delivers heavy traffic regardless of node density and routing topology imbalance, without sacrificing energy consumption. Compared to Orchestra, ALICE delivers 2 times more bidirectional traffic with 98.3% reliability, 70% lower end-to-end latency, 95% less RPL parent changes, and 35% lower radio duty-cycle.

Contributions. The contributions of this work are fourfold.

- We investigate autonomous TSCH cell scheduling, focusing on providing low latency and high reliability for bidirectional traffic delivery regardless of node density and topology imbalance.
- We design ALICE comprising the three elements: (1) directional link-based cell scheduling, (2) multi-channel utilization, and (3) periodic cell schedule change.
- We implement ALICE on Contiki and open the source code, which includes the three features for correct ALICE operation: (1) interaction between RPL and scheduling operation, (2) interaction between RPL and link layer transmission, and (3) Tx cell scheduling *after* packet queueing.



(a) Node distribution with routing topology



(b) Cell scheduling when a slotframe has 36 cells

Figure 1: An example of TSCH operation in a 4-hop, 23-node network where node A is the border router

- We show that ALICE outperforms Orchestra in all aspects: reliability, throughput, latency, routing stability, and duty-cycle. The performance gap becomes larger under heavy traffic load.

2 BACKGROUND

This work investigates the TSCH cell scheduling issue, when TSCH operates under the RPL routing protocol, to provide high reliability, high throughput, and low energy consumption together. This section provides a background for understanding the rest of this paper: TSCH and its scheduling issue, RPL, and LLN application scenarios with heavy traffic.

2.1 TSCH (Link Layer)

TSCH combines time slotted access with channel hopping, standardized as part of IEEE 802.15.4e in 2012 [51]. In a TSCH network, each synchronized node shares its time information by periodically broadcasting Enhanced Beacon (EB). A new node first scans wireless channels and joins a TSCH network when receiving a valid EB and synchronizing its time to that of the EB sender. Each TSCH node has a time source node and re-synchronizes its time to that of the time source node when receiving an EB from the time source.

Slotframe Architecture. Figure 1(b) illustrates TSCH scheduling when routing topology is given by Figure 1(a). By using the time synchronization, TSCH constructs a two-dimensional *slotframe* comprising L_{SF} timeslots (time domain) and L_{CH} channel offsets (frequency domain). L_{SF} is called slotframe length; the same slotframe structure is repeated every L_{SF} timeslots. A pair of a time offset t_o ($0 \leq t_o < L_{SF}$) and a channel offset c_o ($0 \leq c_o < L_{CH}$), i.e., (t_o, c_o), defines a *cell*. The slotframe in Figure 1(b) has $L_{SF} = 9$ and $L_{CH} = 4$, resulting in 36 cells. Each timeslot is long enough for a node to send a packet and receive an ACK (i.e., 10∼15 ms).

Each node relates t_o and c_o to the physical time and frequency channel, respectively, as follows: Each TSCH timeslot has its ID, called Absolute Slot Number (ASN). ASN is set to zero when a network starts, increased by one at the end of each timeslot, and shared by all nodes through EB. Then time offset t_o of the ASN-th timeslot is represented by using modulo operation, as

$$t_o = mod(\ ASN,\ L_{SF}\ ). \tag{1}$$

A TSCH network has a Frequency Hopping Sequence (FHS) consisting of IEEE 802.15.4 channels (usually 4 channels which are known to have the least amount of wireless interference, i.e., $L_{FHS} = 4$). The FHS is shared by all nodes through EB. Then the relationship between c_o and the actual frequency channel is expressed by

$$Channel = FHS(\ mod(\ ASN + c_o,\ L_{FHS}\ )\ ). \tag{2}$$

This slotframe architecture enables to avoid wireless interference and multipath fading by using multiple channels [12].

Cell Scheduling Issue. Each node’s behavior at each timeslot is determined by a *cell scheduling* algorithm: what channel to use and whether to sleep, transmit, or listen. The cell scheduling largely impacts TSCH performance, both packet delivery and energy consumption. Figure 1(b) shows a scheduling example where each cell is empty or filled with node pairs. If a cell (t_o, c_o) is filled with a node pair `(A,B)`, the cell is scheduled for node A to send a packet to node B. Each node wakes up only at the timeslot which has a cell where it is scheduled to do something, and sleeps otherwise.

There are important design criteria for cell scheduling: **(1) Adjacent node pairs should be scheduled in different cells.** Although TSCH uses slotted CSMA/CA to avoid collision when multiple transmissions are scheduled at one cell, a scheduling algorithm should minimize this case. For example in Figure 1(b), cells $(0, 3)$, $(3, 2)$, and $(4, 0)$ are shared by multiple adjacent node pairs, resulting in hidden node collision or CSMA/CA contention. Instead, a scheduling algorithm needs to distribute these adjacent node pairs by utilizing the five empty cells. **(2) A node should not be scheduled for multiple operations in the same timeslot.** In Figure 1(b), node G is scheduled to send at cell $(7, 3)$ and also receive at cell $(7, 2)$, only either of which can happen at once. In addition, Node E is scheduled to send to nodes B, J, and K, all at cell $(6, 1)$, only one of which can happen at once. These cases do not incur collision/contention but degrade both reliability and latency performance. **(3) A node should know its scheduling information by itself, only using local information.** Due to LLN node’s resource constraint, any elegant mechanism which works well in powerful networks, such as LTE and WiFi, can be failed in LLN if it incurs significant control overhead.

2.2 RPL (Routing Layer)

RPL is the IPv6 routing protocol for LLN, standardized in 2012 [29]. RPL is designed to build bidirectional routes between a border router (e.g., node A in Figure 1(a)) and thousands of resource-constrained (possibly battery-powered) nodes, mainly for reliable *upward* traffic delivery. To this end, RPL constructs a Destination-Oriented Directed Acyclic Graph (DODAG) rooted at a border router, based on a distance vector metric from the border router, called RANK; In a DODAG, a node closer to the border router should have a smaller RANK. Routing information including RANK is exchanged/updated by broadcasting DODAG Information Object (DIO) messages. The DIO broadcast period is managed by TrickleTimer to achieve both fast route recovery and low control overhead.

Each node sets its upward route by selecting a preferred parent node according to its Objective Function (OF). OF defines how each node computes its own RANK within a DODAG and selects its preferred parent among the neighbor nodes. Although the definition of OF is decoupled from the main standard, the most commonly used OF is Minimum Rank with Hysteresis Objective Function (MRHOF) [43], which uses ETX to compute RANK by default.

After selecting the preferred parent, a node transmits a Destination Advertisement Object (DAO) message through the upward route, which sets the downward route as the reverse of the upward route (symmetric bidirectional route). When changing the preferred parent due to link variation, a node sends a DAO to the new preferred parent to setup a new downward route and a No-path DAO to the old preferred parent to remove the old downward route. Given that a DAO transmission failure (after link-layer retransmissions) can ruin a downward route, RPL provides an *optional* feature that a parent node replies with a DAO-ACK when receiving a DAO to ensure a bidirectional parent-child relationship. When the DAO-ACK feature is enabled, a node waits for a DAO-ACK after sending a DAO to its preferred parent and resends the DAO at the routing layer when failing to receive a DAO-ACK for a certain timeout period. If the node fails to receive a DAO-ACK after the maximum number of DAO retransmissions, it changes the preferred parent and repeat the procedure until receiving a DAO-ACK.

2.3 The Case of Heavy Traffic in LLN

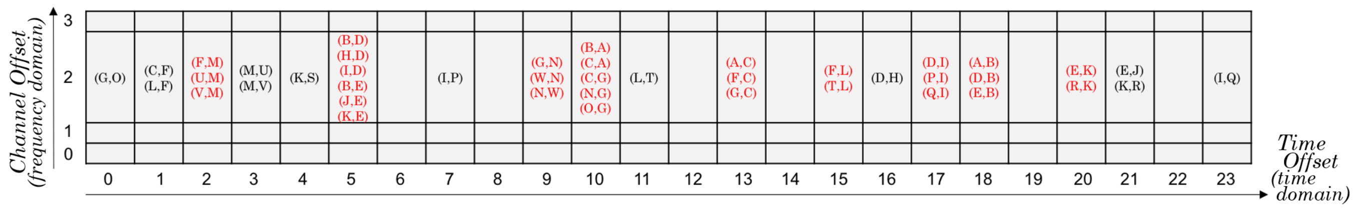
Although traditional LLN applications (e.g., environment monitoring) typically generate low-rate traffic, as LLN’s use cases have been more investigated and diversified, a number of applications require an LLN to provide high throughput.

Large-scale and/or Dense Deployment. Once an application requires a large-scale and/or dense node deployment, nodes near the border router should deliver heavy traffic even though each node generates low-rate traffic. For example, CISCO’s Connected-Grid Mesh for smart grid constructs a large-scale LLN with ~5000 nodes and ~7 hops [45]. Electronic shelf label (ESL) system for smart market needs to build an LLN with ultra-high node density since even a small store typically has thousands of price tags [18]. In addition, the ESL server delivers a visual information to an electronic price tag for a price update [39], which requires heavy traffic delivery.

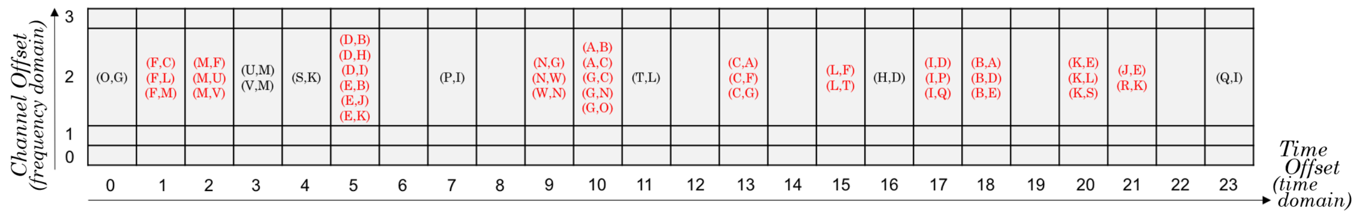
Frequent Data Reporting. Modern IoT applications require frequent data reporting for meaningful data analytics (e.g., by using deep learning). For example, machine health monitoring for smart factory requires a vibration sensor attached on a machine to frequently report its data [13]. The Heating, Ventilation, and Air Conditioning (HVAC) system may include anemometer deployment to diagnose problems in a building and collect air flow measurements for improved HVAC control. An anemometer needs to send a contiguous stream of data to maintain calibration (e.g., 1 Hz sampling rate) [36]. In these applications, the number of serviceable nodes is strictly bounded by LLN’s throughput performance.

3 PRELIMINARY STUDY: ORCHESTRA

To ground our study, we present a preliminary case study of Orchestra [34], the *de facto* cell scheduling method implemented on Contiki. We briefly describe its scheduling mechanism and experimentally analyze its limitations, which motivates our ALICE design.



(a) Unicast slotframe of the receiver-based (O-RB) scheduling



(b) Unicast slotframe of sender-based (O-SB) scheduling

Figure 2: An example of Orchestra cell scheduling with the 23-node topology in Figure 1(a) and $L_{SF}^{UC} = 24$. Node-based scheduling makes multiple adjacent links share the same cell, resulting in contention/collision and/or latency problems.

3.1 Autonomous "Node"-based Scheduling

The key features of Orchestra are its *autonomous* operation and tight *interaction with RPL*, which enable each node to schedule TSCH cells considering routing topology, *by itself*, with *zero* additional overhead. Specifically, each node needs only its MAC address (node ID), parent-child relationship (at the routing layer) for cell scheduling. Orchestra provides three types of slotframes to deliver various traffic in LLN: (1) EB slotframe, (2) broadcast slotframe, and (3) unicast slotframe. To avoid overlapped schedules among the three slotframe types, each slotframe type uses only one fixed channel offset: $c_o = 0$ for the EB slotframe, $c_o = 1$ for the broadcast slotframe, and $c_o = 2$ for the unicast slotframe, respectively.

The EB slotframe is for exchanging EBs. Each node schedules two cells in an EB slotframe, one for transmitting its EB and the other for receiving an EB from its time source. Each node sets its RPL parent node as the time source. When L_{SF}^{EB} is the length of the EB slotframe, node k sends its EB in a fixed cell $(t_o, c_o) = (t_o^{EB,Tx}(k), 0)$, where $t_o^{EB,Tx}(k)$ is a time offset for EB transmission of node k and calculated by using node k's ID, as

$$t_o^{EB,Tx}(k) = mod(\ Hash(ID(k)),\ L_{SF}^{EB}\ ). \tag{3}$$

Note that node ID is *hashed*[1] (i.e., randomized) to mitigate EB collision. Based on Eq. (3), node k determines when to receive an EB from its time source, $t_o^{EB,Rx}(k)$, by hashing the time source's ID; $t_o^{EB,Rx}(k)$ can be changed when the RPL parent node is changed.

The broadcast slotframe is for broadcasting control messages, such as DIO, and transmitting any packet when a unicast cell is not scheduled (as a failover). Orchestra activates only one *fixed* cell, $(t_o, c_o) = (0, 1)$, in a broadcast slotframe, where *all nodes* wake up to support broadcasting. Since broadcast timeslots are scheduled more frequently as the broadcast slotframe length, L_{SF}^{BC}, decreases, L_{SF}^{BC} should be short enough to cope with control packet transmissions.

[1]Note that the choice for $Hash(x)$ is an implementation choice. The modulo function can also be used as $Hash(x)$.

The unicast slotframe is for packet exchanges between dedicated two nodes, which is the subtlest part of cell scheduling. To this end, Orchestra provides two types of scheduling: receiver-based (O-RB) and sender-based (O-SB). In O-RB, each node has only one fixed cell in a unicast slotframe to *receive* packets from any node, e.g., $(t_o, c_o) = (t_o^{UC,Rx}(k), 2)$ for node k. The time offset for node k's receipt of unicast packets, $t_o^{UC,Rx}(k)$, is determined similar to Eq. (3), by hashing node ID as

$$t_o^{UC,Rx}(k) = mod(\ Hash(ID(k)),\ L_{SF}^{UC}\ ) \tag{4}$$

where L_{SF}^{UC} is the unicast slotframe length. Node k's parent and children nodes extract $t_o^{UC,Rx}(k)$ from node k's ID and use the cell $(t_o, c_o) = (t_o^{UC,Rx}(k), 2)$ when sending a packet to node k. Each node's sending cells can be changed as routing topology varies. In contrast, in O-SB, each node has only one fixed cell in a unicast slotframe to *send* packets to any node, e.g., $(t_o, c_o) = (t_o^{UC,Tx}(k), 2)$ for node k. The time offset $t_o^{UC,Tx}(k)$ is determined as Eq. (4). Node k's parent and children nodes listen at the cell $(t_o, c_o) = (t_o^{UC,Tx}(k), 2)$ in preparation for node k's packet transmission. Each node's receiving cells can be changed as routing topology varies.

Given that Orchestra mainly uses node ID, we call it *node-based scheduling*. Due to the randomization of a hash function, each node is likely to have its unique cell for reception (O-RB) or transmission (O-SB) when L_{SF}^{UC} is larger than the number of nodes N. When a node is scheduled for multiple operations in a timeslot, if any, the node chooses an operation with this order: EB, broadcast, and unicast; to give higher priority for control packet exchanges.

3.2 Problems

We claim that node-based scheduling is inefficient. Figures 2(a) and 2(b) show scheduling examples of O-RB and O-SB, respectively, when 23 nodes are distributed/connected as in Figure 1(a). Note that the EB and broadcast slotframes, not shown in Figures 2(a) and 2(b), use channel offset 0 and 1 for active cells, respectively, never colliding with the active cells in the unicast slotframe.

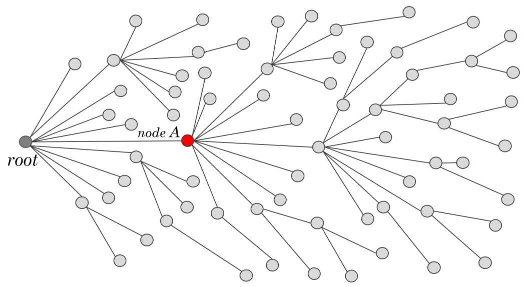


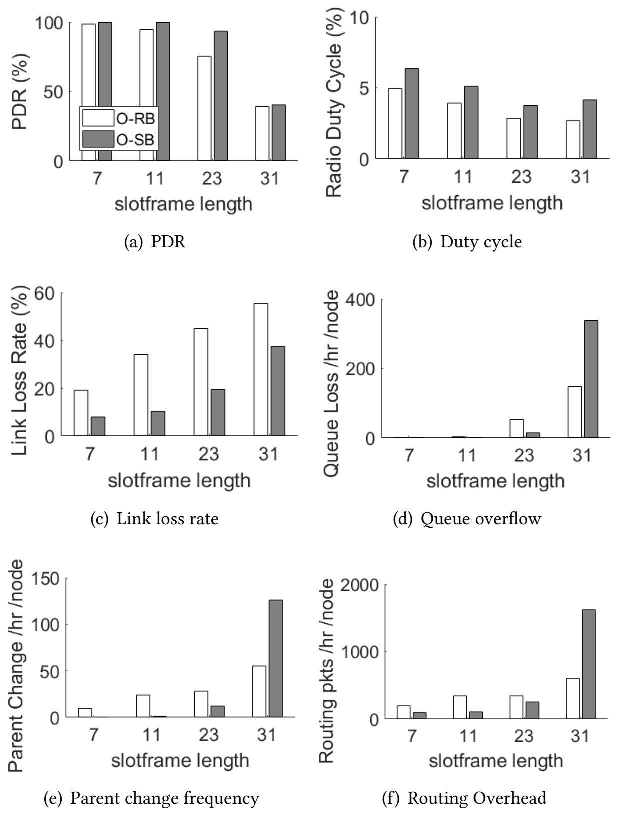
Figure 3: A snapshot of RPL routing topology on the IoT-LAB testbed (Grenoble), with 68 M3 nodes using -17 dBm transmission power.

The unicast slotframe in Figures 2(a) and 2(b) is long enough for each node to have a unique cell, since $L_{SF}^{UC} = 24$. However, both O-RB and O-SB have problems due to *node-based* scheduling. In the case of O-RB, a cell $(t_o^{UC,Rx}(k), 2)$ is scheduled for `(All_Nodes,k)`. If node k has N_k neighbors (parent and children nodes), N_k nodes contend with each other to send packet to node k in one cell, $(t_o^{UC,Rx}(k), 2)$, which causes hidden node collision or CSMA/CA contention. The fact that node k has its unique receiving cell does not mean that it can be free from contention/collision (e.g., the cells $(2, 2)$, $(13, 2)$, $(15, 2)$, $(17, 2)$, $(18, 2)$, and $(20, 2)$ in Figure 2(a)). The problem becomes more severe as a node has more incoming traffic, more children nodes, or an overlapped schedule.

On the other hand, in the case of O-SB, a cell $(t_o^{UC,Tx}(k), 2)$ is scheduled for `(k,All_Nodes)`. This scheduling method does not incur contention/collision, but may cause a latency or queue overflow problem when a node has many packets to send, since it can send only one packet per unicast slotframe: node k needs N_k slotframes to send a packet to each neighbor. When downstream traffic is heavy (e.g., ESL system), the border router, which transmits all downward packets, significantly suffers from this problem. In addition, each node has to wake up and listen to the medium at each neighbor's Tx cell in preparation for receiving any packet (e.g., node k wakes up N_k times per slotframe), which increases radio duty-cycle due to longer idle listening period.

To verify the qualitative analysis, we evaluate O-RB and O-SB (Contiki implementation) on the IoT-LAB public testbed in Grenoble, France. We use 68 embedded devices (M3) with transmission power of -17 dBm where one node acts as the border router, resulting in a 6-hop LLN as shown in Figure 3. We use MRHOF for RPL and 4 channels for TSCH ($L_{FHS} = L_{CH} = 4$). We generate heavy bidirectional traffic, 2 pkts/min of upward traffic from each node and 2 pkts/min downward traffic to each node. In total, the border router is required to deliver 268 pkts/min. Figures 4(a) through 4(f) show various performance metrics of O-RB and O-SB according to the unicast slotframe length L_{SF}^{UC}, when $L_{SF}^{EB} = 397$ and $L_{SF}^{BC} = 19$.

Trade-off about Slotframe Size. Figures 4(a) and 4(b) show that both O-RB and O-SB have a trade-off about the slotframe length. As L_{SF}^{UC} increases, radio duty-cycle is improved but packet delivery ratio (PDR) is significantly degraded (~40% when $L_{SF}^{UC} = 31$). This is because Orchestra utilizes less cells for transmission/reception as L_{SF}^{UC} increases, losing more packets while reducing energy consumption. Figures 4(c) and 4(d) show that both O-RB and O-SB suffer more link loss and queue loss as L_{SF}^{UC} increases, verifying



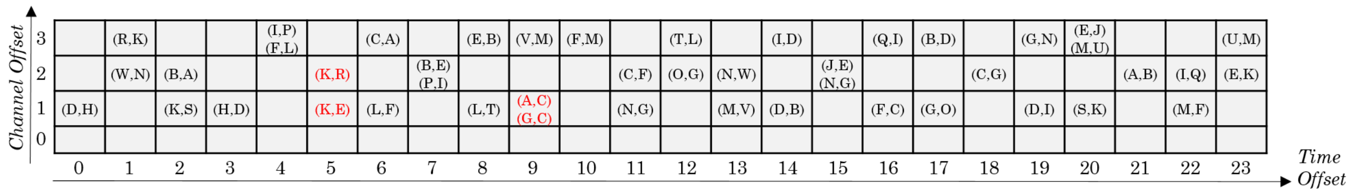
(a) PDR (b) Duty cycle (c) Link loss rate (d) Queue overflow (e) Parent change frequency (f) Routing Overhead

Figure 4: Various performance metrics of Orchestra RB (O-RB) and SB (O-SB) according to unicast slotframe length under bidirectional traffic.

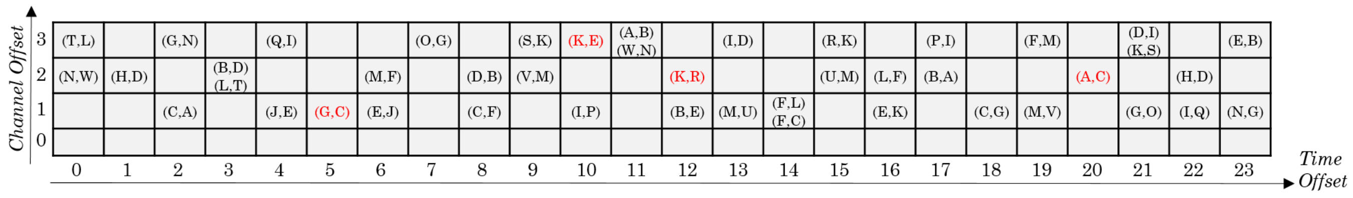
that Orchestra experiences significant contention/collision/delay problems with a large unicast slotframe size due to lack of transmission/reception opportunities.

Note that Orchestra's unicast slotframe size needs to be larger than the number of nodes to allocate a unique cell for each node (an Rx cell in O-RB, a Tx cell in O-SB). It is a reasonable assumption that the unicast slotframe size of 68 may be the optimum. When $L_{SF}^{UC} > 68$, Orchestra may waste many cells without transmission/reception. When $L_{SF}^{UC} < 68$, Orchestra cannot preserve a unique cell for each node. However, Figure 4(a) shows that the PDR performance of O-RB and O-SB starts to be degraded when the unicast slotframe size becomes larger than 7 and 11, respectively, both of which are much smaller than 68. This shows that in Orchestra under heavy traffic, the loss of utilizing less cells is more significant than the gain of providing a unique cell for each node. Therefore for reliable delivery of heavy traffic with stable routing, Orchestra needs to use a small unicast slotframe size, which sacrifices energy consumption.

Interaction between RPL and TSCH. Figures 4(e) and 4(f) show that the routing layer is significantly affected by the performance degradation at the link layer. When L_{SF}^{UC} is large, RPL tries to change the preferred parent more frequently to avoid link loss, which is helpless since the link loss comes from contention/collision rather than bad link quality. Therefore RPL's effort ends up with nothing but churning topology and increasing control overhead (e.g., DIO, DAO, and DAO-ACK). Furthermore, RPL has a load balancing



(a) ALICE schedule example at a unicast slotframe



(b) ALICE schedule example at the next unicast slotframe (reflecting time varying cell scheduling)

Figure 5: An example of ALICE scheduling with the 23-node topology in Figure 1(a) and $L_{SF}^{UC} = 24$. Its multi-channel utilization and directional link-based scheduling minimize contention/collision without sacrificing latency. In addition, time varying scheduling prevents specific links from suffering persistent contention/collision.

problem [17]. Figure 3 verifies that RPL provides a significantly unbalanced routing topology where one red node (node A) has to deliver traffic from/to 44 nodes (66% of total traffic). This topology imbalance intensifies contention/collision at the link layer; TSCH scheduling must be improved to deliver heavy traffic.

Comparison between O-RB and O-SB. While having similar behavior as described above, O-RB and O-SB operate differently in details. Figures 4(c) and 4(d) show that O-RB incurs more link loss while O-SB incurs more queue loss. This matches our qualitative analysis: O-RB causes multiple senders contend in a same cell, which results in link loss due to contention/collision. On the other hand, O-SB provides only one transmission opportunity for each node per slotframe, which increases latency and queue loss.

Figures 4(a) and 4(b) show that when using the same L_{SF}^{UC}, O-SB provides higher PDR with more radio duty cycle due to more idle listening. If we allow O-RB and O-SB to use different unicast slotframe sizes and target high PDR (>99%), O-RB should use $L_{SF}^{UC} = 7$ while O-SB should use $L_{SF}^{UC} = 11$. If we compare the two cases, O-RB and O-SB provide comparable PDR and duty-cycle performance. However, O-SB provides more stable routing topology, with less parent switches and routing packets. This is because, as shown in Figure 4(c), O-SB has less link loss, which mitigates RPL's misunderstanding of link quality. This demonstrates that if a scheduling method has a trade-off between link loss and queue loss, it should first resolve the link loss for a better interaction with RPL.

On the other hand, is it possible to escape from this trade-off: link loss (contention/collision) vs. queue loss (latency)?[2] We claim that it is possible if we change the scheduling paradigm, *from node-based scheduling to link-based scheduling*. This is what ALICE is about, which is described in the next section.

[2]We claim that this trade-off is **a fundamental limitation of Orchestra**. To verify this, we also evaluated a multi-channel version of Orchestra, which does not show meaningful performance improvement (figures are omitted for brevity). This is because the performance degradation of Orchestra comes from its inefficient use of existing resource, not from lack of resource.

4 ALICE: AUTONOMOUS DIRECTIONAL "LINK"-BASED CELL SCHEDULING

Our preliminary study showed that Orchestra has performance issues even when providing a unique cell for a node, since multiple adjacent links are scheduled in the same cell. To resolve the problem, our intuition is that a unique cell should be allocated for *each directional link*. This section presents the design and implementation aspects of our proposed scheduling method, called *ALICE*.

4.1 Overview

ALICE follows the basic architecture of Orchestra: (1) the same types of slotframes (EB, broadcast, and unicast) with the same scheduling priority, (2) autonomous scheduling, and (3) interaction with layer 3. However, *ALICE schedules cells in the unicast slotframe differently*, (1) using directional link rather than node ID, (2) utilizing multiple channel offsets, and (3) changing cell allocation at every slotframe.

These changes can be made very easily. Specifically, assuming that node k is scheduled to send a packet to node l in a cell $(t_o, c_o) = (t_o^{UC}(k,l), c_o^{UC}(k,l))$, the time offset $t_o^{UC}(k,l)$ and the channel offset $c_o^{UC}(k,l)$ for unicast communication over the directional link (k,l) are calculated, respectively, as

$$t_o^{UC}(k,l) = mod(\ Hash(\alpha ID(k) + ID(l)),\ L_{SF}^{UC}\ ) \tag{5}$$

$$c_o^{UC}(k,l) = mod(\ Hash(\alpha ID(k) + ID(l)),\ L_{CH} - 1\ ) + 1. \tag{6}$$

Here the coefficient α is used to differentiate traffic directions,[3] e.g., link (k,l) vs. link (l,k). As in Eq. (6), when $L_{CH} = 4$, ALICE utilizes channel offsets 1, 2, and 3 for the unicast slotframe.

ALICE does not require any additional information for cell scheduling compared to Orchestra, enabling a node to autonomously allocate a unique cell for each directional link including the node. In addition, each node's Tx and Rx cells are changed as routing topology varies. When N nodes are connected through RPL (DODAG topology), the number of directional links is $2N - 2$. Given that

[3]We use $\alpha = 256$ (maximum value of node ID, the last byte of MAC address).

ALICE utilizes three channel offsets, it can allocate a unique cell for each directional link when the unicast slotframe size is larger than $(2N-2)/3$. Then, each node sends a packet to (or receives a packet from) any neighbor node in a unique cell.

Since a hash function can return the same value for different links, ALICE may allocate one cell for multiple links. To resolve the issue, ALICE changes cell schedules every unicast slotframe, which prevents specific links from being overlapped forever. To this end, we define Absolute SlotFrame Number (ASFN) as $ASFN = \lfloor ASN/L_{SF}^{UC} \rfloor$ where $\lfloor x \rfloor$ is the floor function. Then the time and channel offsets for the directional link (k,l) in the $ASFN$-th unicast slotframe can be calculated as

$$t_o^{UC}(k,l,ASFN) = mod(\ Hash(\alpha ID(k) + ID(l) + ASFN),\ L_{SF}^{UC}\ ) \tag{7}$$

$$c_o^{UC}(k,l,ASFN) = mod(\ Hash(\alpha ID(k) + ID(l) + ASFN),\ L_{CH} - 1\ ) + 1. \tag{8}$$

Figure 5 shows an example of ALICE scheduling when 23 nodes are distributed/connected as in Figure 1(a). Figure 5(a) shows that ALICE utilizes multiple channels and allocates a unique cell for each directional link. This almost nullifies the contention, collision, and latency problems. As exceptional cases, (K,R) and (K,E) are scheduled at the same timeslot (different cells), and (A,C) and (G,C) are scheduled even at the same cell. Although these exceptions can occasionally happen, the problem is likely to be solved at the next unicast slotframe due to the $ASFN$-based scheduling. For example, Figure 5(b) shows the cell schedules at the next unicast slotframe, where all the previously overlapped links are distributed.

4.2 Design and Implementation

We implement ALICE on Contiki 3.0 and open the code.[4] While changing the node-based scheduling to the link-based scheduling is straightforward (i.e., some formula changes), careful implementation choices should be made for time varying scheduling.

Time Varying Scheduling. At each unicast sloftrame, ALICE changes cell schedules completely. To this end, when a node finishes the operation at the last active cell (for either Tx or Rx) of the $ASFN$-th unicast slotframe, it increases $ASFN$ by one and reallocates cells by using Eqs. (7) and (8). To this end, we use a simple 32-bit integer mix function [55] for $Hash(x)$. Modern embedded hardware performs this computation fast enough (e.g., with an M3 node computing power, getting (t_o, c_o) of 1,000 links takes only 1 ms) to not affect the next timeslot operation.

Packet-Cell Matching *on the fly*. Orchestra determines what cell to use for a packet transmission, when inserting the packet into the queue; each packet in the queue has a *fixed* Tx cell. When a node wakes up at the cell (t_o, c_o) where it is scheduled to send something to some node, the node checks if its packet queue has any packet whose Tx cell is (t_o, c_o), and sends the packet (i.e., Tx cell-based packet search). Note that Orchestra has a fixed relationship between a packet’s MAC destination and its Tx cell.

In contrast, ALICE changes a Tx cell for the same MAC destination at each unicast slotframe. Since it is possible for a packet’s Tx cell to be changed while the packet is queued, setting each packet’s Tx cell before queueing can ruin the forwarding procedure. To resolve the problem, we implement ALICE to set a packet’s Tx cell *on the fly*. Specifically, a node inserts a packet in the queue *without setting its Tx cell*. When node k wakes up at the cell (t_o, c_o) where it is scheduled to send something to node l, node k checks if its packet queue has any packet whose next hop (MAC destination) is node l, and sends the packet (i.e., next hop-based packet search).

Early Packet Drop. Given that Orchestra fixes a packet Tx cell before queueing it, a node tries to send a packet to its MAC destination at its determined Tx cell, even when the MAC destination is no longer a neighbor. Sending a packet to a non-neighbor node mostly fails even after link layer retransmissions, which only wastes energy/time resource and increases contention/collision.

In contrast, when an ALICE node searches the packet queue based on the MAC destination information at each Tx cell, the node checks if each packet’s MAC destination is still its RPL neighbor. When a packet’s destination is no longer a neighbor due to physical link variation, the node drops the packet since it cannot schedule a Tx cell for the packet (the packet may not be sent forever). This packet drop procedure prevents a hopeless packet from occupying the packet queue space for a long time.

Interaction with RPL. ALICE’s cell scheduling for a link (k,l) operates well only when both nodes k and l know that they are valid RPL neighbors. Given that a node selects the preferred parent by itself, it always knows the parent information. On the other hand, a node cannot know a new child information until it receives a DAO from the child node. Therefore, when a node fails to transmit a DAO to its parent, ALICE scheduling can fail.

To address the issue, we enable RPL’s DAO-ACK option. Specifically, when a node selects a new parent, it changes the preferred parent to the new parent only when successfully sending a DAO to the new parent and receiving a DAO-ACK from the new parent. On the other hand, a node adds a new child node when receiving a DAO and removes an old child node when receiving a no-path DAO. During a parent change procedure, before setting a complete new bidirectional parent-child relationship, the DAO, no-path DAO, and DAO-NACK messages are scheduled to be sent through the broadcast slotframe.

Putting it All Together. To implement the above features, we provide three main components for ALICE: (1) `ALICE_Operation_Manager`, (2) `ALICE_Packet_Selector`, and (3) `ALICE_Next_Cell_Scheduler`.

The `ALICE_Operation_Manager` contains ALICE’s main operation loop, as described in Algorithm 1. At the start of an active cell, the manager figures out what operation is required for the current cell: Tx and/or Rx. Note that it is possible that both Tx and Rx are required for a single cell due to a schedule overlap. In this case, the manager prefers Tx operation. If Tx is required, the manager executes the `ALICE_Packet_Selector` to get a packet to send in the current cell. If the `ALICE_Packet_Selector` returns a valid packet, the manager sends the packet in the current cell. If the manager gets no packet from the `ALICE_Packet_Selector`, it sleeps (Tx only cell) or listens (both Tx/Rx cell) to the medium in the scheduled cell. On the other hand, if only Rx is required, the manager listens to the medium in the current cell. After finishing the required operation,

[4]https://github.com/skimskimskim/ALICE

the manager executes the `ALICE_Next_Cell_Scheduler` to get the next active cell and sleeps until then.

Algorithm 1: ALICE Operation Manager

```
1  $(t_o, c_o) \leftarrow$ Initialize TSCH and set the current cell;
2  while (1) do
3      operation ← get_operation($(t_o, c_o)$);
4      if operation has TX then
5          packet ← ALICE_packet_selector($(t_o, c_o)$, $ASFN$);
6          if packet is not NULL then
7              transmit_packet(packet,channel($c_o$, $ASN$));
8          else if operation has RX then
9              listen(channel($c_o$, $ASN$));
10     else if operation has RX then
11         listen(channel($c_o$, $ASN$));
12     Add/remove cells (whenever RPL topology is updated);
13     $(t_o, c_o) \leftarrow$ ALICE_next_cell_scheduler($t_o$, $ASFN$);
14     sleep_until($(t_o, c_o)$);
```

Algorithm 2: ALICE Packet Selector

```
   input  : cell $(t_o, c_o)$, $ASFN$
   output : packet
1  for pkt in PacketQueue do
2      dest ← get_next_hop(pkt);
3      if dest is not in RPL_NEIGHBOR_LIST then
4          remove pkt from PacketQueue;
5      else
6          $(t_o^{pkt}, c_o^{pkt}) \leftarrow$ ALICE_get_TX_cell(dest, $ASFN$);
7          if $(t_o^{pkt}, c_o^{pkt}) == (t_o, c_o)$ then
8              packet ← pkt;
9              terminate;
10 packet ← NULL;
```

Algorithm 2 describes how the `ALICE_Packet_Selector` works. The selector receives a cell information as an input and returns one (or zero) packet to send in the cell. To this end, it searches all packets in the packet queue. If a packet's next hop node is not a RPL neighbor, the selector removes the packet from the queue ('Early Packet Drop'). Otherwise, the selector looks into the current cell scheduling status to check what cell is scheduled for sending to the next hop node. If the found Tx cell is equal to the input cell, the selector returns the packet ('Packet-Cell Matching on the fly'). If no packet's Tx cell matches the input cell, it returns NULL.

Algorithm 3 describes the `ALICE_Next_Cell_Scheduler`'s operation. The scheduler searches the active cell list and finds the active cell scheduled for the nearest future in the current unicast slotframe. If it fails to find a valid next cell, the current cell is the last active cell in the current unicast slotframe. Then, the scheduler reschedules cells for the next unicast slotframe by increasing $ASFN$ by one ('Time Varying Scheduling').

5 EVALUATION

We evaluate the effectiveness of ALICE on the IoT-LAB [8] with the same configuration as in Section 3.2. For comparison, we also evaluate both O-RB and O-SB of Orchestra.

Algorithm 3: ALICE Next Cell Scheduler

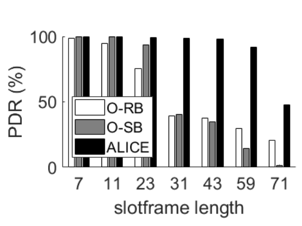
```
   input  : time offset $t_o^{cur}$, $ASFN$
   output : next cell $(t_o^{nxt}, c_o^{nxt})$
1  $(t_o^{nxt}, c_o^{nxt}) \leftarrow NULL$ ;
2  for cell $(t_o, c_o)$ in Unicast_Slotframe_Cell_List do
3      if $(t_o^{nxt}, c_o^{nxt}) == NULL$ and $t_o > t_o^{cur}$ then
4          $(t_o^{nxt}, c_o^{nxt}) \leftarrow (t_o, c_o)$;
5      else
6          if $t_o > t_o^{cur}$ and $t_o < t_o^{nxt}$ then
7              $(t_o^{nxt}, c_o^{nxt}) \leftarrow (t_o, c_o)$;
8          else if $t_o == t_o^{nxt}$ then
9              $(t_o^{nxt}, c_o^{nxt}) \leftarrow$ get_higher_priority_cell($(t_o, c_o)$,
                 $(t_o^{nxt}, c_o^{nxt})$);
10 if $(t_o^{nxt}, c_o^{nxt}) == NULL$ then
11     ALICE_Unicast_SF_Scheduler($ASFN$+1) ;
12     $(t_o^{nxt}, c_o^{nxt}) \leftarrow$ ALICE_next_cell_scheduler(-1, $ASFN$);
```

5.1 Impact of Unicast Slotframe Size

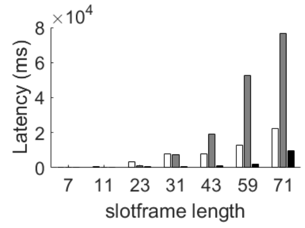
Figures 6(a) through 6(f) show various performance metrics of O-RB, O-SB, and ALICE according to the unicast slotframe size, when delivering bidirectional traffic (2 pkts/min from each node and 2 pkts/min to each node).

Figures 6(a) and 6(b) show that the packet delivery performance of O-RB and O-SB is significantly degraded as L_{SF}^{UC} increases, confirming the results of Section 3.2. Specifically, as L_{SF}^{UC} increases, O-SB's latency performance is degraded, much more severely than the O-RB case, due to lack of transmission opportunity. On the other hand, ALICE maintains good packet delivery performance even when L_{SF}^{UC} is large. Thus ALICE outperforms both O-RB and O-SB in all L_{SF}^{UC} values and the performance gain becomes more significant as L_{SF}^{UC} increases. For example, when $L_{SF}^{UC} = 43$, ALICE provides 2.5 times better PDR and 83-93% lower latency than O-RB and O-SB. This verifies that directional link-based scheduling and time varying scheduling mechanisms efficiently utilize the unicast slotframe space, resolving the contention/collision issue of O-RB and the latency issue of O-SB, simultaneously.

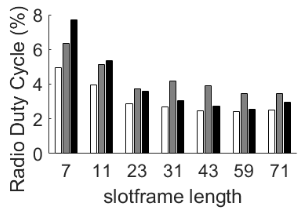
More importantly, Figure 6(c) shows that ALICE obtains the above performance gain without sacrificing energy consumption. When routing topology is stable, ALICE incurs more radio duty cycle than O-RB and O-SB since ALICE utilizes more channels and cells (e.g., when $L_{SF}^{UC} = 7$). However, as L_{SF}^{UC} increases, O-RB and O-SB suffer significant link loss and queue loss, respectively, as shown in Figures 6(d) and 6(e). The link loss coming from contention/collision (O-RB) makes RPL misunderstands that physical link quality is bad. The queue loss coming from latency (O-SB) drops many routing packets at the queue. Therefore, both O-RB and O-SB fail to provide stable routing topology with a large L_{SF}^{UC} value, which causes many parent changes (Figure 6(f)) and routing packet overhead. This inefficient operation significantly increases radio duty cycle of O-RB and O-SB, making ALICE provide duty cycle performance comparable to O-RB and better than O-SB when L_{SF}^{UC} is large.



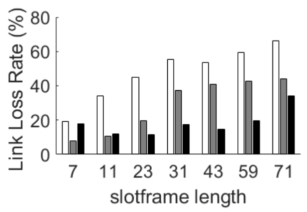
(a) PDR



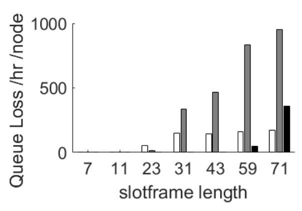
(b) Latency



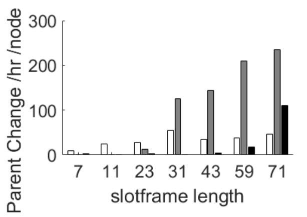
(c) Duty cycle



(d) Link loss rate



(e) Queue loss



(f) Parent change frequency

Figure 6: Performance of Orchestra and ALICE according to unicast slotframe length under bidirectional traffic (2 pkts/min from each node and 2 pkts/min to each node). ALICE provides high PDR and routing stability even with large slotframe size, significantly improving duty cycling.

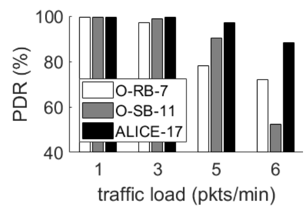
When allowing each scheme to use different L_{SF}^{UC} and targeting the high PDR (>99%), the best unicast slotframe sizes for O-RB, O-SB, and ALICE are 7, 11, and 23, respectively. When comparing these three cases, ALICE provides comparable PDR to O-RB and O-SB, while improving latency and duty cycle.

5.2 Impact of Traffic Load

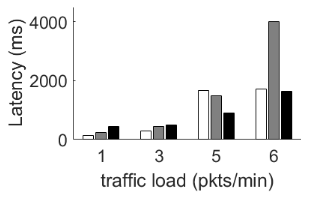
Next, we investigate the impact of traffic load on each scheme's operation. Based on the results in Section 5.1, we use the unicast slotframe size 7, 11, and 17 for O-RB, O-SB, and ALICE, respectively. Figures 7(a) through 7(f) show various performance metrics when traffic load varies from 1 pkts/min to 6 pkts/min.

Figure 7(a) shows that the PDR performance of both O-RB and O-SB is rapidly degraded as traffic load increases, even though their unicast slotframe sizes are small enough. On the other hand, ALICE maintains high PDR with heavy traffic load, even with larger unicast slotframe size compared to O-RB and O-SB. This is because ALICE provides more transmission opportunities than O-SB and less contention/collision than O-RB.

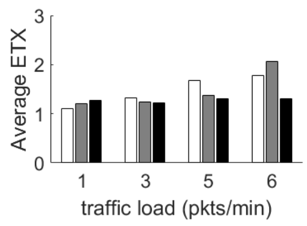
Figure 7(b) shows that ALICE provides slightly longer latency than O-RB and O-SB under light traffic load due to a larger unicast slotframe size. However, as L_{SF}^{UC} increases, ALICE incurs better latency performance than the others. Figure 7(c) shows that ALICE maintains low ETX under heavy traffic load, while both O-RB and



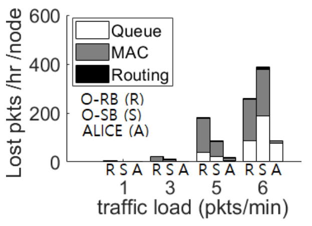
(a) PDR



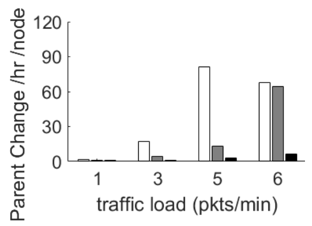
(b) Latency



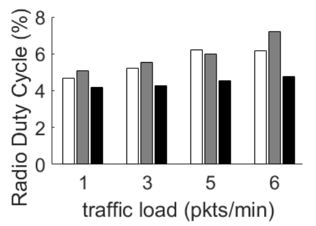
(c) ETX



(d) Lost packet



(e) Parent change frequency



(f) Duty cycle

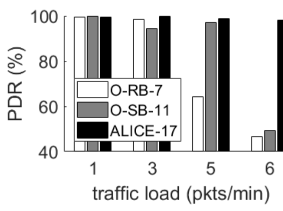
Figure 7: Performance of Orchestra and ALICE according to traffic load when O-RB, O-SB, and ALICE use slotframe size 7, 11, and 17, respectively.

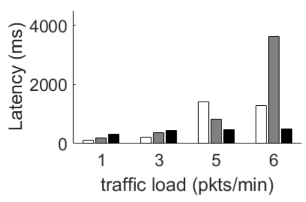
O-SB incurs higher ETX. Figure 7(d) shows that ALICE significantly reduces all types of packet loss compared to O-RB and O-SB. Specifically, when the traffic load is very high, ALICE's packet losses are mostly at the queue instead of the link, which enables RPL to provide stable topology (Figure 7(e)). Again, these results verify that ALICE addresses all the contention, collision, and latency problems efficiently. Lastly, Figure 7(f) shows that ALICE achieves better packet delivery performance even with lower duty cycle than the others; ALICE outperforms Orchestra in all aspects.

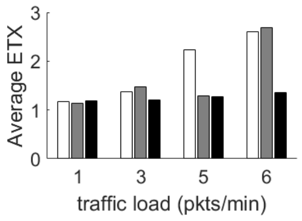
5.3 Impact of Node Density

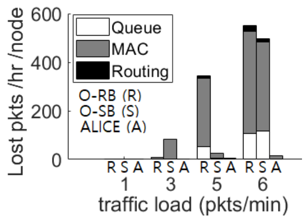
Now we investigate the impact of node density. To this end, we increase transmission power from -17 dBm to 3 dBm (shorter and denser routing topology) and perform the same experiments as in Section 5.2. Figure 8(a) through 8(f) show the results. When comparing these results with those of Figures 7(a) through 7(f), it clearly shown that the performance of Orchestra worsens with high node density. In contrast, interestingly, the performance of ALICE becomes even better. Therefore, the performance gap between Orchestra and ALICE becomes more significant.

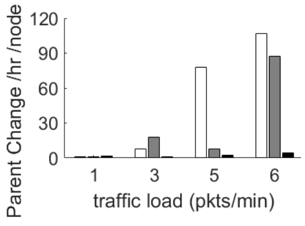
We revisit the scheduling methods to discuss the reason. O-RB allocates only one Rx cell for each node in a unicast slotframe, where it receives packets from *all neighbors*. With high node density (many neighbors), O-RB suffers contention/collision even more. Similarly, O-SB allocates only one Tx cell for each node, where it sends packets


(a) PDR


(b) Latency


(c) ETX


(d) Lost packet


(e) Parent change frequency

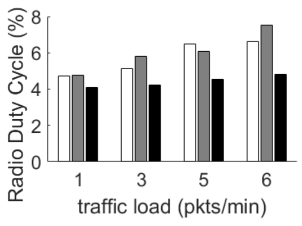

(f) Duty cycle

Figure 8: Performance of Orchestra and ALICE according to traffic load when O-RB, O-SB, and ALICE use slotframe size 7, 11, and 17, respectively. Transmission power is 3 dBm for all the 68 nodes.

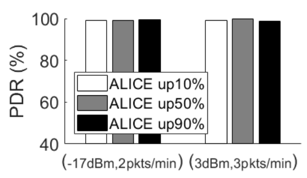
to all neighbors, intensifying the latency problem as a node has more neighbor nodes. Our results show that O-RB suffers from high node density more than O-SB, resulting in worse PDR performance (Figure 8(a)) due to severe link loss (Figure 8(d)).

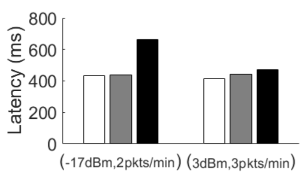
In contrast, ALICE provides one cell for each directional link, regardless of node density. Thus, ALICE does not lose anything with high node density but has the benefit of a shorter routing distance. As shown in Figures 8(b) and 8(d), ALICE provides better latency with significantly reduced queue loss when node density increases, verifying that it takes advantage of shorter routing distance. Overall, under a dense node deployment and a heavy traffic load (6 pkts/min), compared to Orchestra, ALICE achieves 2 times more throughput with 98.3% reliability, 70% lower latency, 95% less parent changes, 35% lower duty cycle.

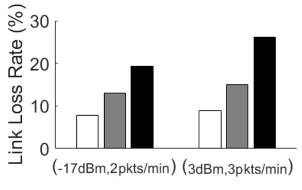
5.4 Impact of Traffic Pattern

We have verified ALICE's superiority over Orchestra through various experiments. Going further, we now focus more on ALICE's intrinsic behavior and limitation. To this end, we evaluate ALICE ($L_{SF}^{UC} = 17$) in various bidirectional traffic patterns: while maintaining the total traffic load, we change the ratio of upward traffic to downward traffic (i.e., 1:9, 5:5, and 9:1). The results are plotted in Figures 9(a) through 9(d).

Figure 9(a) shows that ALICE maintains high PDR regardless of traffic patterns, verifying its robustness. However, it does show


(a) PDR


(b) Latency


(c) Link loss rate

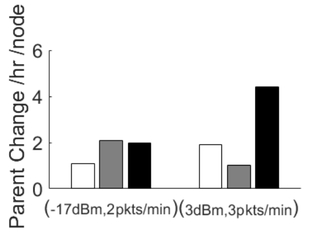

(d) Parent change frequency

Figure 9: Performance of ALICE according to traffic pattern, when $L_{SF}^{UC} = 17$. We include two different combinations of traffic load and node density.

somewhat performance degradation as upward traffic becomes dominant, longer latency (Figure 9(b)), more link losses (Figure 9(c)), and more parent changes (Figure 9(d)). This is because ALICE allocates *only one cell* for every directional link. Although this approach is better than Orchestra, it still has a weakness when a specific link has to deliver much more traffic than other links. For example, when delivering upward packets, a node receives packets from many children nodes (multiple links) and sends all the packets to its parent node (one link); While a node receives multiple upward packets in a unicast slotframe, it can send only one packet to the parent in a unicast slotframe. Our results show that ALICE may need to use a smaller unicast slotframe to support upward-focused traffic than downward-focused traffic.

We believe that the further way to go is to dynamically allocate more cells for a link delivering more traffic without breaking the autonomous scheduling nature: *autonomous scheduling based on both directional link and traffic load.*

6 RELATED WORK

Due to a strong requirement of "simple operation", initial link layer protocols in multihop LLN are mostly asynchronous and use a fixed single channel [27][52]. However, using a single channel provides limited reliability on the 2.4 GHz ISM band, which is notoriously lossy due to multipath fading and interference [1, 10, 11, 21, 26, 30, 33]. To alleviate the problem, some groundwork was done, which reveals the potential of time synchronization and channel hopping. Pister and Doherty designed TSMP which synchronizes nodes in a multihop LLN within a few hundred microseconds [49]. Watteyne *et al.* experimentally evaluated frequency channel hopping with CTP, *defacto* multihop collection protocol, showing that channel hopping provides more robust multihop connectivity than using a single channel [40]. After TSCH [51] was standardized as part of IEEE 802.15.4e in 2012 [50], a number of studies have investigated this protocol. Some of them focus on TSCH cell scheduling which can be classified into three categories: (1) centralized, (2) distributed, and (3) autonomous approaches.

Centralized Cell Scheduling. Centralized methods make a root node gather the network information from all nodes and perform cell scheduling for each node. TASA [23] provides a centralized cell scheduling which reduces latency and radio duty-cycle based on global tree topology and each node's traffic load information. With the same information, MODESA [37] focuses on load balancing among frequency channels. Although these approaches provide theoretically optimal scheduling, in practice, they suffer significant control overhead to update network information for the root node and slow schedule adjustment when topology is changed. For this reason, centralized scheduling methods are more suitable for a static environment where routing topology rarely changes [32].

Distributed Cell Scheduling. Distributed methods try to allocate cells through a handshaking procedure among neighbors. DeTAS [25] is the distributed version of TASA, where a parent node gathers all children nodes' upstream traffic load and schedules when/where it receives packets from each child node. However, in DeTAS, the root should first gather the entire upstream traffic information and allocate cells for its one-hop children nodes. Then the one-hop children nodes perform cell scheduling for their children nodes. This top-down scheduling propagation cannot quickly adjust to link dynamics. Wave [31] tried to minimize uplink delay in a decentralized scheduling method, which also incurs the top-down scheduling propagation. In MPLS [7], nodes exchange path and reservation messages for proper bandwidth reservation. To calculate required bandwidth, each node should have knowledge on its subtree and 2.5 layer service (multiprotocol label switching) is used for distributed slotframe scheduling, inevitably causing traffic overhead. D-MSR [56] makes each node use a handshaking mechanism to allocate common unused cells for communicating with a new TSCH neighbor. Another mechanism [24] proposed a negotiation protocol to provide on-the-fly bandwidth reservation for slotframe scheduling.

Autonomous Cell Scheduling. Orchestra [34] is the first autonomous TSCH cell scheduling method, which is proven to operate on resource-constrained nodes. It enables each node to schedule its own cells by using RPL neighbor information without any traffic overhead. After the advent of Orchestra, several autonomous TSCH cell scheduling mechanisms have been proposed.

Escalator [48] tried to reduce latency for upstream traffic delivery. To this end, each node schedules not only its own cells but also the cells for all of its subtree nodes. The scheduling aims to provide this property: if a node sends an upward packet to its parent node in a cell, the parent node should be able to send the packet in the *right next cell* (the packet goes upwards as an escalator). However, it only focuses on upward traffic not allocating any unicast cell for downward traffic. Moreover, a node suffers more for idle listening overhead as its subtree grows and the method has not yet been evaluated on a large testbed. e-TSCH-Orch [38] also tried to reduce latency of Orchestra focusing on collection (upstream) scenario. Based on Orchestra schedule, when a node sends a packet to a neighbor node, it indicates the number of packets in its transmission queue on the packet footer. Its neighbor node schedules that amount of consecutive Rx cells and the sender cleans its transmission queue quickly. However, this simple strategy ruins Orchestra's original schedule, worsening the contention and collision problems, and has not yet been evaluated on a large testbed. We emphasize that using a proper evaluation methodology is necessary, particularly in this LLN regime, to incorporate environmental challenges [9, 15].

Differently from the previous works focusing on the unicast transmission, Vallati *et al.* modified the broadcast slotframe of Orchestra [54]. The authors focused on RPL's unique behavior: a lot of control packets are generated during the network initialization period or the topology change period, but control packets are moderately generated during the rest of the time. Based on the observation, the authors dynamically schedule multiple cells in the given broadcast slotframe size, providing more broadcast cells only when many control packets should be delivered. This strategy reduces radio duty cycling without sacrificing routing stability. This approach and ALICE are complementary to each other: the former handles the broadcast slotframe and the latter handles the unicast slotframe. Combining these two methods can be a future work.

Novelty of ALICE. The TSCH scheduling research has shifted from centralized to distributed and autonomous methods. Orchestra is groundbreaking as the first autonomous scheduling method considering routing information. After Orchestra, some autonomous scheduling techniques have been proposed, none of which escape from the Orchestra's node-ID hashing-based scheduling scheme.

However, recall that a transmission does not happen on a node but a *directional link*, a pair of sender and receiver nodes. What then is the right thing for a TSCH cell scheduler to do, as a *link layer* protocol? It must be able to handle each directional link independently. Focusing on this fundamental principle, ALICE uses **directional link**, rather than node, as the identifier for autonomous cell scheduling. In addition, ALICE adopts a **time-varying scheduling technique**, which resolves cell collision among multiple directional links, a representative problem of autonomous scheduling. These design choices are realized by our **careful implementation of tight interaction between layer 2 and layer 3**. As a result, ALICE outperforms Orchestra **without violating the nature of autonomous scheduling**.

To the best of our knowledge, ALICE is **the first autonomous cell scheduling which performs better than Orchestra in *all aspects***: reliability, throughput, latency, routing stability, and energy consumption. In contrast, Escalator and e-TSCH-Orch, even in their own ideal simulation scenarios, do sacrifice energy consumption to improve latency performance.

Looking forward, we believe that ALICE paves the "right" way for more advanced cell scheduling techniques in that it provides a **fundamental structure** for directional link-based autonomous scheduling (i.e., handling each directional link separately). Building on this structure, adaptive scheduling techniques considering various information, such as real-time traffic demand, can be developed.

7 CONCLUSION

In this paper, we have systematically investigated the autonomous cell scheduling problem in TSCH. Our preliminary study of Orchestra revealed the limitations of node-based autonomous scheduling, such as contention, collision, and latency, which waste resource and result in a significant performance degradation under heavy traffic load. We designed and implemented ALICE, a novel directional link-based autonomous scheduling method. On the open IoT-LAB testbed with 68 nodes, the effectiveness of ALICE was extensively evaluated and compared with Orchestra. Our results verify that

ALICE outperforms Orchestra in all aspects: reliability, throughput, latency, routing stability, and energy consumption. The advantage of using ALICE becomes more significant when traffic load is heavy. Our study proves that ALICE has the potential to support a variety of applications, including not only traditional sensor network applications generating low-rate traffic but also emerging heavy traffic, streaming applications. In addition, ALICE provides robust performance with unbalanced routing topology, regardless of node density, which can support various deployment scenarios. Overall, with its open implementation, ALICE can serve as a new *de facto* cell scheduling method for TSCH.

ACKNOWLEDGMENTS

This work was supported by IITP MSIT (No.2015-0-00557), MSIP ITRC (IITP-2018-2015-0-00378), NRF MSIP (No.2016R1A5A1012966), Institute for Industrial Systems Innovation of Seoul National University, and the Dept. of Energy Grant No. DE-EE0007685, California Energy Commission, Intel Corporation. We are thankful to the anonymous reviewers and the shepherd for their valuable feedback.

REFERENCES

[1] G. Anastasi, M. Conti, and M. Di Francesco. 2011. A comprehensive analysis of the MAC unreliability problem in IEEE 802.15.4 wireless sensor networks. *IEEE Trans. Indus. Inf.* 7, 1 (2011), 52–65.

[2] ANSI/ISA. 2011. ANSI/ISA-100.11a-2011 Wireless systems for industrial automation: Process control and related applications. *ISA* (2011).

[3] E. Buckland, M. Ranken, M. Arnott, and P. Owen. August 5, 2016. IoT Global Forecast & Analysis 2015-25, Strategy Report. *Machina Research* (August 5, 2016).

[4] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. 2004. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *IEEE LCN*. 455–462.

[5] Atis Elsts et al. 2016. Microsecond-Accuracy Time Synchronization Using the IEEE 802.15. 4 TSCH Protocol. In *IEEE LCN Workshops*. 156–164.

[6] Alan Mainwaring et al. 2002. Wireless sensor networks for habitat monitoring. In *International workshop on Wireless sensor networks and applications*. ACM, 88–97.

[7] A. Morell et al. May 2013. Label switching over IEEE 802.15.4e networks, Transactions on Emerging Telecommunications Technologies. In *Trans. on Emerging Telecommunications Technologies*, Vol. 24. 458–475.

[8] C. Adjih et al. 2015. FIT IoT-LAB: A large scale open experimental IoT testbed. *IEEE World Forum on Internet of Things (WF-IoT)* (2015).

[9] C. A. Boano et al. 2018. IoTBench: Towards a Benchmark for Low-power Wireless Networking. In *CPSBench 2018*.

[10] C. Kishore Singh et al. 2008. Performance evaluation of an IEEE 802.15.4 sensor network with a Star Topology. *ACM Wireless Networks* 14, 4 (2008), 543–568.

[11] D. De Guglielmo et al. 2016. Accurate and efficient modeling of 802.15.4 unslotted CSMA-CA through event chains computation. *IEEE Trans. Mobile Comput.* (2016).

[12] D.D. Guglielmo et al. 2016. IEEE 802.15.4e: a Survey. In *Computer Communications*, Vol. 88. 1–24.

[13] Deokwoo Jung et al. 2017. Vibration Analysis for IoT Enabled Predictive Maintenance. In *IEEE ICDE*. 1271–1282.

[14] David Stanislowski et al. 2014. Adaptive synchronization in IEEE802. 15.4 e networks. *IEEE Trans. on Industrial Informatics* 10, 1 (2014), 795–802.

[15] H.-S. Kim et al. 2017. Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey. *IEEE Commun. Surv. Tutor* 19, 4 (2017), 2502–2525.

[16] Hyung-Sin Kim et al. 2017. Do not lose bandwidth: Adaptive transmission power and multihop topology control. In *IEEE DCOSS*. 99–108.

[17] H.-S. Kim et al. 2017. Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks. *IEEE Transactions on Mobile Computing* 16, 4 (2017), 964–979.

[18] H.-S. Kim et al. 2017. Smarter markets for smarter life: applications, challenges, and deployment experiences. *IEEE Communications Magazine* 55, 5 (2017), 34–41.

[19] Hyung-Sin Kim et al. 2018. System architecture directions for post-soc/32-bit networked sensors. In *ACM SenSys*. 264–277.

[20] JeongGil Ko et al. 2010. MEDiSN: Medical emergency detection in sensor networks. *ACM Trans. on Embedded Computing Systems (TECS)* 10, 1 (2010), 11.

[21] K. Yedavalli et al. 2008. Enhancement of the IEEE 802.15.4 MAC protocol for scalable data collection in dense sensor networks. *Proceedings of WiOPT* (2008).

[22] M. Mohammad et al. 2016. Oppcast: Exploiting Spatial and Channel Diversity for Robust Data Collection in Urban Environments. *IPSN* (2016).

[23] Maria Rita Palattella et al. 2012. Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. (2012).

[24] M. R. Palattella et al. 2016. On-the-fly bandwidth reservation for 6tisch wireless industrial networks. *IEEE Sensors* 16, 2 (2016), 555–560.

[25] Nicola Accettura et al. 2013. DeTAS: A decentralized traffic aware scheduling technique enabling IoT-compliant multi-hop low-power and lossy networks. *Second IEEE WoWMoM Workshop on IoT-SoS*, 337–350.

[26] P. Di Marco et al. 2014. Modeling IEEE 802.15.4 networks over fading channels. *IEEE Trans. on Wireless Communication* 13, 10 (2014), 5366–5381.

[27] P. Joseph et al. 2004. Versatile low power media access for wireless sensor networks. In *ACM SenSys*. 95–107.

[28] Pangun Park et al. 2011. Breath: An adaptive protocol for industrial control applications using wireless sensor networks. *IEEE Trans. on Mobile Computing* 10, 6 (2011), 821–838.

[29] P. Thubert et al. March 2012. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *IETF, RFC 6550* (March 2012).

[30] R. Daidone et al. 2014. On evaluating the performance impact of the IEEE 802.15.4 security sub-layer. *Computer Communications* 47 (2014), 65–76.

[31] Ridha Soua et al. 2015. Wave: a distributed scheduling algorithm for con- vergecast in IEEE 802.15.4e TSCH networks. *Trans. on Emerging Telecommunications Technologies* 27, 4 (2015), 557–575.

[32] R. Tavakoli et al. 2018. Topology Management and TSCH Scheduling for Low-Latency Convergecast in In-Vehicle WSNs. *IEEE Trans. Ind. Informatics* (2018).

[33] S. Brienza et al. 2013. Strategies for optimal MAC parameter setting in IEEE 802.15.4 wireless sensor networks: A performance comparison. *IEEE ISCC* (2013).

[34] Simon Duquennoy et al. 2015. Orchestra: Robust mesh networks through autonomously scheduled TSCH. *ACM SenSys* (2015).

[35] S. Kim et al. 2007. Health Monitoring of Civil Infrastructures using Wireless Sensor Networks. In *IPSN*. ACM, 254–263.

[36] Sam Kumar et al. 2018. TCPlp: System Design and Analysis of Full-Scale TCP in Low-Power Networks. *arXiv preprint arXiv:1811.02721* (2018).

[37] Soua Ridha et al. 2012. MODESA: an optimized multichannel slot assignment for raw data convergecast in wireless sensor networks. *IEEE IPCCC* (2012).

[38] S. Rekik et al. 2018. Autonomous and traffic-aware scheduling for TSCH networks. In *Computer Networks*, Vol. 135. 201–211.

[39] Taewon Suh et al. 2018. Electronic Shelf Lables: Prototype Development and Validation Using A Design Science Approach. *Journal of Information Technology Management* 29, 4 (2018), 23.

[40] T. Watteyne et al. 2009. Reliability through frequency diversity: why channel hopping makes sense. In *the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM, 116–123.

[41] Thomas Watteyne et al. 2012. OpenWSN: a standards-based low-power wireless development environment. *Trans. on Emerging Telecommunications Technologies* 23, 5 (2012), 480–493.

[42] T. Watteyne et al. 2015. Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement. *IETF RFC 7554* (2015).

[43] O. Gnawali and P. Levis. September 2012. The Minimum Rank with Hysteresis Objective Function. *IETF, RFC 6719* (September 2012).

[44] IEC. April 27, 2010. Industrial Communication Networks Wireless Communication Network and Communication Profiles WirelessHART. *IEC 62591 Ed. 1.0 b:2010* (April 27, 2010).

[45] Cisco Systems Inc. Open. Connected Grid Networks for Smart Grid - Field Area Network / CG-Mesh. http://www.cisco.com/web/strategy/energy/field_area_network.html. (Open).

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

[47] R. Musaloiu-E and A. Tezis. 2008. Minimizing the effect of wifi interference in 802.15.4 wireless sensor networks. *Intl. Journal of Sensor Networks* (2008).

[48] S. Oh, D. Hwang, K. Kim, and K. Kim. April, 2018. Escalator: An Autonomous Scheduling Scheme for Convergecast in TSCH. In *Sensors*, Vol. 18(4). MDPI, 1–25.

[49] K Pister and Lance Doherty. 2008. TSMP: Time synchronized mesh protocol. *IASTED Distributed Sensor Networks* (2008), 391–398.

[50] IEEE Computer Society. 2006. IEEE standard for information technology, Part 15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs). *IEEE* (2006).

[51] IEEE Computer Society. 2012. IEEE standard for information technology, 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. *IEEE Computer Society* (2012).

[52] Y. Sun, Omer G., and D. B Johnson. 2008. RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Conference on Embedded network sensor systems*. ACM, 1–14.

[53] P. Thubert. March 2012. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). *IETF, RFC 6552* (March 2012).

[54] C. Vallati, S. Brienza amd G. Anastasi, and S. Das. April, 2018. Improving network formation in 6TiSCH networks. In *Trans. on Mobile Computing*, Vol. 1. IEEE, 1–13.

[55] Thomas Wang. Open. 32-bit Integer Hash Function. https://gist.github.com/badboy/6267743. (Open).

[56] P. Zand, A. Dilo, and P. Havinga. June 2013. D-MSR: A Distributed Network Management Scheme for Real-Time Monitoring and Process Control Applications in Wireless Industrial Automation. *Sensors* (June 2013).

View publication stats