공학박사 학위논문

Autonomous Link Scheduling for Time Slotted Channel Hopping Networks

타임 슬롯 채널 호핑 네트워크를 위한 자율적 링크 스케쥴링

2019 년 8 월

서울대학교 대학원

전기.컴퓨터 공학부

김 서 향

Autonomous Link Scheduling for Time Slotted Channel Hopping Networks

타임 슬롯 채널 호핑 네트워크를 위한 자율적 링크 스케쥴링

지도교수 김 종 권

이 논문을 공학박사 학위논문으로 제출함

2019 년 4 월

서울대학교 대학원

전기·컴퓨터 공학부

김 서 향

김 서 향의 공학박사 학위논문을 인준함

2019 년 6 월

위 원 장	전 화 숙	
부위원장	김 종 권	
위 원	박 세 웅	
위 원	권 태 경	
위 원	최 재 혁	

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. We also extend ALICE and propose TRF-ALICE. In this extended method, the real-time traffic load on each link is reflected on TSCH scheduling in *autonomous* way. Thereby, each directional

link can allocate multiple cells per slotframe to support increased traffic load. We also implement and evaluate TRF-AILCE on IoT-LAB public testbed and prove that the proposed method improves the performance of ALICE by increasing throughput by more than 2 times and reducing latency by 50% while showing a similar amount of radio duty cycle.

Keywords: 802.15.4e, TSCH, Scheduling, Sensor Network, Internet of Things
Student Number: 2013-20761

Contents

List of Figures

Chapter 1

Introduction

Wireless Sensor Network (WSN) has been being studied for tens of years until now [2][3][4]. During this time, communication and hardware technologies have been developed and data rates and calculation speed have been much increased. Now, WSNs are connected to the Internet and make our life more comfortable providing various services [5][6][7]. However, wireless channel is less reliable compared to wired channel. Moreover, wireless devices' lifetime is dependent on their battery. To develop more reliable and efficient wireless communication techniques, various research groups are being created.

IEEE [8] and IETF [9] are the leading global research societies providing global specifications. Recently, IEEE standardized 802.15.4e [10] specification to support highly reliable and energy efficient wireless communication service targeting industrial automation, and it is also investigated and standardized by IETF 6TiSCH working group [11]. With this global research trend, this thesis[1] provides autonomous link scheduling method for reliable wireless communica-

[1]This thesis includes our recent work [1] https://doi.org/10.1145/3302506.3310394.

tion technique focusing on IEEE 802.15.4e and 6TiSCH compliant service.

Unlike wired communication, the wireless communication environment is vulnerable. It suffers from multipath fading, interference and limited battery [12][13]. Wireless medium is continuously changing over time and trivial changes on wireless environment such as people's movement, furniture arrangement, temperature and humidity directly affect the status of medium [14][15]. Due to a cost problem, commercial IoT network uses constrained nodes with low RAM size under 10kB [16][17]. In this vulnerable lossy wireless environment, constrained hardware should provide low power and reliable communication service.

A network comprised of many resource-constrained embedded devices on lossy environment is called as Low power and lossy network (LLN). This network aims to support various applications, such as environment monitoring [18], factory automation [19], smart hospital [6], and smart market [7]. 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 [74]. For example, IETF standardized 6LoWPAN in 2007 [40] and IPv6 Routing Protocol for LLNs (RPL) in 2012 [56], 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 [46][53][29], 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) [10] as part of the IEEE 802.15.4e standard [51]. It provides synchronous node wake-up for

low energy consumption and frequency channel hopping for reliable communication over lossy wireless links [59]. Due to its potential, TSCH has received a significant attention from both industry and academia: WirelessHART [33] and ISA100.11a [22] 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 [26][43] and OpenWSN [61], 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 3). The centralized approaches make the border router gather network information and schedule TSCH cells for all nodes [55][30], 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 [60][24][27][39], 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 [41], an *autonomous* cell scheduling method considering RPL rout-

ing 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 4).

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 [34]) and frequent data-gathering IoT applications (e.g., machine or structure health monitoring with acceleration and vibration [37][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 [36][75], 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 5).

We implement ALICE on Contiki and evaluate its performance on IoT-LAB [76] (an open LLN testbed) with 68 embedded devices (Section 6). 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.

We also extend ALICE to reflect the *real-time traffic load* per each *directional link* in *autonomous* way (Section 7). This extended version of ALICE named TRF-ALICE does not require any traffic overhead to exchange information of real-time traffic load. We introduce the method that the receiver can autonomously estimate the transmitter's the number of transmission attempt per slotframe. Based on *the average number of transmission attempt per slotframe*, the transmitter and the receiver of one directional link *autonomously* decide the number of tx/rx cells per slotframe to allocate. We also implement TRF-ALICE on Contiki and evaluate the performance on IoT-LAB public testbed (Section 8). We prove that TRF-ALICE improves the performance of ALICE by increasing throughput by more than 2 times and reducing latency by 50% while showing a similar amount of radio duty cycle.

Contributions. The contributions of this work are fivefold.

- 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.

- 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.

- We extend ALICE by considering the *real-time traffic load* in *autonomous* way. We also show that the extended method TRF-ALICE improves the performance much more than ALICE in terms of PDR, latency and radio duty-cycle.

Chapter 2

Background Knowledge

In wireless network, to communicate between a node pair, a node should transmit a radio signal through a shared wireless medium. This wireless medium can be interpreted with two dimensional time-frequency domain. If multiple nodes transmit radio signal at the same time at the same channel, it results in collision. To avoid unwanted events such as collision or interference, we need well designed Medium Access Control (MAC) protocol.

IEEE 802.15.4 standard [66] is one of the representative low power communication techniques. It was designed targeting Low-Rate Wireless Personal Area Networks (LR-WPANs) and used widely to construct energy efficient wireless sensor networks. However, this protocol supported communication on a single channel. If there are continuous interference on the used channel, the network on this protocol could not support reliable end-to-end communication service. With a single channel, multiple node pairs cannot communicate simultaneously decreasing efficiency on resource utilization. Moreover, multipath fading is common problem on wireless communication, which can be mitigated by using chan-

nel hopping technique [12]. With these drawbacks, it could not support highly reliable communication service that is requested to support 99.9% packet delivery ratio.

To develop reliable service for industrial network, protocols such as WirelessHART [33] and ISA100.11a [22] have been designed utilizing channel hopping technique. With the increasing demand for reliable and energy efficient communication service, IEEE have constructed 802.15.4e [10] group and amended 802.15.4 mac protocol targeting industrial automation service.

Time Slotted Channel Hopping (TSCH) is one of mac modes provided in IEEE 802.15.4e standard and it was also selected as the main protocol of Internet Engineering Task Force (IETF) 6TiSCH working group [11]. With this reason, TSCH is the currently most widely used protocol regarding 802.15.4e standard these days. TSCH is a mac protocol that is on the basis of time synchronization and channel hopping techniques. Due to its strong characteristics providing both reliability and efficiency, it is also used on various purposes such as smart health care service platform on residential house [5], in-vehicle network [69], and smart grid system [70]. In addition, because TSCH is completely open not requiring license fees, some researchers have decided to change the protocol with TSCH to easily innovate their technologies during the projects [68][5].

IETF also specified the routing protocol for low power and lossy networks named RPL [56] by Roll working group [71]. To implement 6TiSCH compliant network, RPL should be used as a routing protocol [60] [67]. With this reason, the recent research trend regarding 802.15.4e or industrial network implements both RPL and TSCH together [27] [42] [48]. Though TSCH provides highly reliable and energy efficient communication service, the specification does not answer us how to schedule each node on time-frequency domain.

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 [10]. In a TSCH network, each synchronized node shares its time information by periodically broadcasting Enhanced Beacon (EB). At the first time, a TSCH network is created by the TSCH PAN (Personal Area Network) Coordinator. It creates the network and periodically broadcasts EB. In the EB, all the information needed to join this network is written. Since TSCH is time slotted network, EB should provide Absolute Slot Number, called ASN. When the PAN coordinator creates the network, ASN is initialized to 0 and the value is increased by one at the end of each time slot.

TSCH is based on channel hopping technique. So all the nodes in the network should share the Frequency Hopping Sequence (FHS). FHS can be written in the EB or the network can assume that all the nodes in the network already know the used FHS. Before joining the network, general nodes do not have information on TSCH network. So any node who wants to join the TSCH network should listen the wireless channel with scanning mode. In the scanning mode, a node scans the wireless channel with already known Join Hopping Sequence. After a node receives an EB, it sets the EB sender as its TSCH time source and finally joins the network.

In TSCH network, time source is used to synchronize the time between nodes. Each node sets one node as its time source and synchronizes its time

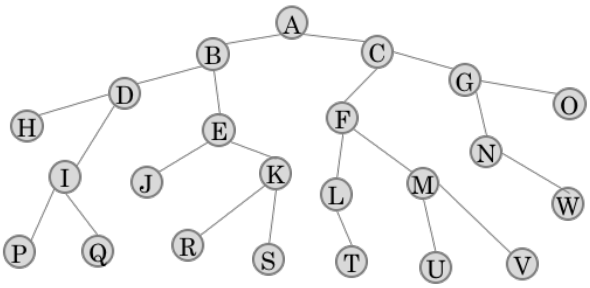
to its time source. As a result, all the nodes in the network synchronize their time to the TSCH PAN coordinator. For this, EB is used. Each node should receive EB periodically from its time source to synchronize its time to its time source. It also need to broadcast EB periodically since it can be also used as time source of another node.

If one node detects that its ASN is different from its time source, it assumes that it is not synchronized to the TSCH network and leaves the TSCH network. A node who is not associated with a TSCH network should operate in the scanning mode to receive EB from any node who is already associated with a TSCH network. A node who is disassociated cannot transmit any packet. Communication in a TSCH network is based on time synchronization and channel hopping. The node activity just follows two dimensional schedule table which is called as slotframe in TSCH. Following this schedule table, a node transmits the packet, listens the channel, or just sleep saving energy.

Slotframe Architecture. Figure 2.1(b) illustrates TSCH scheduling when routing topology is given by Figure 2.1(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 2.1(b) has $L_{SF} = 9$ and $L_{CH} = 4$, resulting in 36 cells. In each time slot, a node pair can exchange only one data packet and its corresponding ACK. A general duration of each time slot is 10-15 ms.

Each node already knows the current ASN value since it is a shared number in the TSCH network through the periodically exchanged EBs. By using ASN,



(a) Node distribution with routing topology

Channel Offset (frequency domain)	0	1	2	3	4	5	6	7	8
3	(B,A) (C,A)	(G,O)	(P,I)		(Q,I)	(D,I)	(F,C)	(G,C)	
2	(L,T) (U,M)	(I,Q)		(B,D) (I,D) (H,D) (J,E)	(C,F)	(F,L)	(R,K)	(O,G)	(M,F) (K,S)
1	(N,G)	(B,A)	(D,B)	(B,E) (I,P)	(K,E)	(S,K)	(E,B) (E,J) (E,K)	(K,R) (W,N)	(F,M)
0	(G,N)	(V,M) (N,W)	(D,H)	(M,V)	(T,L) (M,U)	(J,E)		(L,F)	

Time Offset (time domain)

(b) Cell scheduling when a slotframe has 36 cells

Figure 2.1 An example of TSCH operation

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

the current time offset can be calculated. If the current ASN value is ASN, and the slotframe length is L_{SF}, the current timeslot t_o can be calculated by using modulo operation, as

$$t_o = mod(\ ASN,\ L_{SF}\ ). \tag{2.1}$$

Channel offset does not mean the physical channel. It is used to calculate the current channel. To calculate the current channel, 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., $L_{FHS} = 4$) together with ASN and channel offset should be used. The FHS is shared by all nodes through EB. Then the relationship between c_o and the actual frequency channel is expressed by

$$Channel = FHS(\ mod(\ ASN + c_o,\ L_{FHS}\ )\ ). \tag{2.2}$$

This slotframe architecture enables to avoid wireless interference and multipath fading by using multiple channels [31].

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 2.1(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 should be periodically activated for communication between a node pair A and B. Since the scheduled slotframe repeats over time, this cell periodically repeats over time enabling their communication. Whenever this cell is activated, a node A can transmit data packet to node B. Since a node B does not know whether a node A will

transmit a packet or not in this cell, it should listen the channel whenever this cell is activated to receive packet from node A successfully. In case of node A, if it has a packet to node B, it just transmits the packet at the corresponding time and channel since it knows the fact that a node B will listen at the same time and channel. If a node A has no packet to transmit to node B, it can just sleep saving energy.

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 2.1(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 2.1(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. To build optimal schedule, a central node could collect global network information from all the nodes in the network. However, if there are topology change on a routing layer, a central node should re-collect the global information, re-schedule

the slotframes of all the nodes in the netework, and re-distribute the updated schedule over the network, which requires significant control traffic overhead and long delay. This centralized scheduling approach can only be used for a fixed and highly restricted network which is unusual.

2.2 RPL (Routing Layer)

RPL is the IPv6 routing protocol for LLN, standardized in 2012 [56]. RPL is designed to build bidirectional routes between a border router (e.g., node A in Figure 2.1(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 Trickle Timer 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) [82]. OF defines how each node computes its own RANK within a DODAG and selects its preferred parent among the neighbor nodes [55] [30]. Although the definition of OF is decoupled from the main standard, the most commonly used OF is Minimum Rank Objective Function with Hysteresis (MRHOF) [30], which uses ETX (Expected Transmission Count) for RANK calculation.

RPL network is initiated by a border router called root node. After creating the network, it periodically broadcast DIO where information needed to join

this RPL network is written. When a general node receives DIO, it joins RPL network by setting this DIO sender as its RPL preferred parent. A node who wants to solicit DIO can broadcast DODAG Information Solicitation (DIS). If a node who already joined one RPL network receives DIS, it replies with DIO to give information on its RPL network. If DIS sender receives DIO, it can join RPL network by setting DIO sender as its preferred parent.

Since nodes in the RPL network periodically broadcasts DIO including its RANK information, any node can receive multiple DIOs from multiple one hop neighbors with different RANK values. Since RANK can be interpreted as the relative cost to reach the root node, if a node finds DIO with lower RANK value, it can change its preferred parent to lower the path cost to the root. With this reason, the topology is continuously changed over time.

Whenever a node selects its 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).

RPL provides two different mode: storing mode and non-storing mode. In storing mode, each node stores routing information to each descendant in its sub network. In non-storing mode, each node does not have any knowledge on its children. Only the root node remains the routing table to all the nodes in the network. As a result, storing mode requires more memory usage to store routing information. In non-storing mode, a node sends unicast DAO to the root node since only the root node remains the routing table. However, in storing mode, a node sends unicast DAO to its preferred parent and a node who received DAO from its children forwards this message to its preferred parent. This process is repeated until the DAO message reaches to the root node. Thereby, all the nodes in the path between the DAO initiator and the root can add routing

information regarding this DAO initiator.

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. A node who received No-path DAO removes routing information regarding this DAO initiator. This message is also forwarded until it reaches the root and all the nodes between the DAO initiator and the root can remove the corresponding routing entry.

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 disabled, DAO is sent in best-effort manner. As a result, if DAO receiver does not store the DAO sender in its routing table due to memory problem, reliable downstream path cannot be established [16].

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.

A node who received DAO from unknown node can allow this node as its child or not. If it allows this node as its child, it sends back DAO-ACK. However, if it has no memory to add this node as its child or other problems, it cannot add this node as its child. In this case, it sends back DAO-NACK to let that node know the situation. In this case, according to the implementation, it finds new preferred parent to get its DAO in or just do nothing setting the DAO-

NACK sender as its preferred parent continuously. By enabling DAO-ACK option, DAO sender can receive feedback from DAO receiver, which enables to establish reliable end-to-end downstream path [16].

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 [34]. 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 [7]. In addition, the ESL server delivers a visual information to an electronic price tag for a price update [65], 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 [35]. 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) [73]. In these applica-

tions, the number of serviceable nodes is strictly bounded by LLN's throughput performance.

Chapter 3

Related Work

For wireless communication, we use wireless medium. Unlike wired medium, wireless medium is open and shared by multiple devices. Including people's natural movement, furniture arrangement, temperature and humidity, any trivial changes surrounding wireless medium can affect the performance of communication [14][15]. Especially, 2.4 GHz ISM band is shared by multiple protocols such as 802.15.4, BLE and Wi-Fi. Due to a strong requirement of "simple operation", initial link layer protocols in multihop LLN are mostly asynchronous and use a fixed single channel [47][54]. 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 [32, 23, 62, 50, 21, 25, 38, 12]. 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 [46]. 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 [59]. After TSCH [10] was standardized as part of IEEE 802.15.4e in 2012 [51], a number of studies have investigated this protocol. Although this protocol was designed for industrial automation and provides high reliability with energy efficiency, this protocol only defines the node's operation at the link layer, but does not answer how to schedule each node's channel access by scheduling the slotframe. Scheduling the slotframe of each node is the same as determining when each node should access and communicate at which channel. As a result, it directly affects network performance. With this reason, slotframe schedule became one of the main research topic regarding TSCH. Existing scheduling methods can be classified into three categories: (1) centralized, (2) distributed, and (3) autonomous approaches.

3.1 Centralized Cell Scheduling.

With a centralized scheduling approach, only a central node schedules the slotframes of all the nodes in the network. Firstly, the central node collects network information from all the nodes. Taking global view on a network, centralized scheduler schedules the slotframes of each node. Since the central node has all information required for slotframe schedule, it can build optimal schedule based on global information. After finishing the schedule, it distributes the scheduled slotframe to each node. When there are topology changes such as new node join event, a central scheduling agency should re-collect all network information again and re-schedule the slotframe based on the recent network information. Updated slotframes should be re-distributed over the network. Because only the central node enforces all decisions about the slotframe schedule, this scheduling approach suffers from slow responsiveness for sudden topology change. As a re-

sult, it is not appropriate to be used on a network with time-varying topology. Moreover, it requires a lot of traffic overhead and long delay in updating the schedule table. So this approach is recommended for the network with fixed topology that requires optimal slotframe schedule.

TASA [44] and MODESA [49] are the centralized schedulers. These two methods are similar except only a few differences. Both methods have very strong assumptions: traffic load per each node and the routing topology do not change over time. Moreover, they assume that the central node knows complete topology information including routing topology, traffic load of each node, conflict relations among nodes. These methods have been designed targeting information collection scenario where only multi-point-to-point traffic exists, they schedules upstream links only. Moreover, they assumed no message loss with 0% link error rate in their performance evaluation.

Based on global information, TASA reduces end to end latency and radio duty-cycle. MODESA uses more information and focuses on load balancing among frequency channels. In MODESA, it considers queue-length of each node and assumes that the root node can have multiple radio interfaces. With this strong assumptions, MODESA could build collision-free schedule. They evaluated the proposed methods by using simple simulation. Since they assume that there is no message loss with 0% link error rate, 100% packet delivery ratio was achieved.

Recently, there was another trial to use centralized scheduling approach on in-vehicle network [69]. In this TSCH network, 31 sensor nodes were deployed in a car creating dense topology. Even though star topology can be used, they used k-ary tree with height two to reduce end-to-end latency. They focused on low latency and used short slotframe length resulting high radio duty cycle. This method was evaluated on a real environment in a car. However, to obtain

global information at a central node, it requires to have *link quality identification phase*. During this phase, each node estimates the link quality between the node itself and its neighbor nodes. Root node should receive this information from all the nodes before conducting scheduling operation.

Although these centralized scheduling approaches provide near-optimal schedule, they suffer from significant control overhead to make the central node to obtain the global network information. Moreover, it takes long time to update the schedule when topology is changed. For this reason, centralized scheduling methods are more suitable for a static environment where routing topology rarely changes.

3.2 Distributed Cell Scheduling.

In distributed scheduling methods, scheduling agency is distributed over all the nodes in the network. Each node negotiates with their neighbor nodes to build their own slotframe schedule. Though scheduling agent is distributed over the network, in some methods [20][52], only a central node can initiate a schedule updating process and each node's slotframe schedule is still dependent on its parent's choice. In these methods, a central node initiates the schedule and the schedule propagates from the root to the leaf nodes. In other methods [45][72], each node builds its own slotframe schedule regardless of its parent's choice. Any node can initiate its slotframe schedule update process and just negotiates with its neighbor nodes. With this approach, it shows quick responsiveness to a sudden topology change. Compared to a centralized approach, it shows higher flexibility and scalability with relatively low control traffic overhead for slotframe schedule.

DeTAS [20] and Wave [52] are the early schedulers using distributed schedul-

ing approach. Though the load of scheduling at the central node has been distributed to the all nodes in the network, only the centralized node can initiate the start of the schedule of all the nodes in the network and each node's schedule is strongly dependent on the schedule of its parent node. As a result, any topology or traffic change results in a scheduling change of the entire network as centralized schedule does.

Both methods assume that each node has a constant traffic load and multipoint-to-point scenario. The root node initiates the start of the scheduling process. The node closer to the root selects cells first. The cell selection rule is simple. Each node selects timeslots those were not selected by its parent.

In more detail, DeTAS [20] 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.

The only difference between DeTAS and Wave is that, DeTAS selects channel offset on the basis of hop-count, and Wave considers conflicting nodes group on its channel offset schedule. Conflicting node means interfering node. To have knowledge on each node's conflicting node group, Wave requires more control packets to collect and exchange further information. Moreover, wave also tries to minimize uplink delay in a decentralized scheduling method, which also incurs the top-down scheduling propagation.

However, these methods can not be considered to be completely distributed scheduling method in that each node's schedule strongly depends on its parent's schedule.

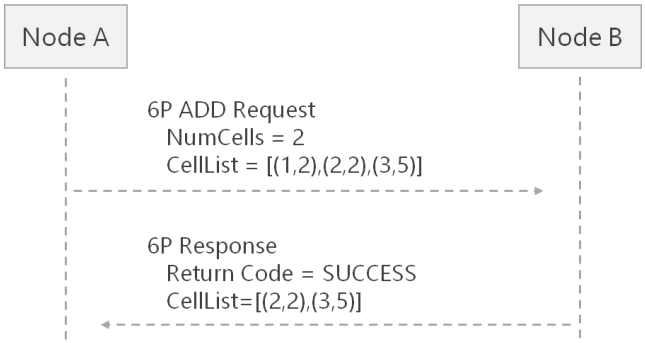


Figure 3.1 Example of 6P transaction

Example of 6P transaction. After the transaction, node A adds two more transmission cells to node B and node B adds two more listen cells to node A by using cells (2,2) and (3,5).

Zand *et al.* pointed out that the previous scheduling methods have drawbacks in terms of slow responsiveness to changes in topology or real-time traffic load and proposed Distributed Management Scheme for Real-time applications (D-MSR) [63]. This novel method makes each node use a handshaking mechanism to allocate common unused cells for communicating with a new TSCH neighbor.

Morell *et al.* introduced 2.5 layer service multiprotocol label switching (MPLS) into slotframe schedule in LLNs [64]. In this method, nodes exchange path and reservation messages for proper bandwidth reservation. To calculate required bandwidth, each node should have knowledge on its subtree and use 2.5 layer service MPLS for distributed slotframe scheduling, inevitably causing further traffic overhead.

Recently, IETF 6TiSCH working group standardized a negotiation protocol

called 6P [72]. By using 6P transaction, two neighboring nodes can add, delete or relocate cells in their TSCH slotframe for their communication.

Let's look at an example of negotiation process using 6P transaction as described in Figure 3.1. Node A wants to add two cells to transmit packet to node B. Then, it sends 6P ADD request to node B. In the message, node A says that it wants to add two more cells and the available cell list is (1,2), (2,2), and (3,5). When node receives this request, it selects two cells among the list and sends back 6P response message to node A. Return code of response message is SUCCESS and the selected two cells are (2,2) and (3,5). After this transactions, node A can send message by using these cells and node B will listen at these cells to receive packets from node A.

Though 6P protocol gives simple and easy methods for negotiation process for slotframe schedule, it is just used as a negotiation protocol. When to trigger 6P transaction and which cells to select are determined by scheduling functions such as OTF [45], SF0 [80] and MSF [81].

These protocols are negotiation-based distributed scheduling methods that use 6P transaction. With these distributed schedulers, each node flexibly schedules its own slotframe based on the current traffic load. Whenever the required traffic load changes between a node pair, they can add or delete cells based on the negotiation process. Generally, to avoid frequent schedule changes, they use threshold when changing the number of cells allocated between a node pair.

OTF [45] schedules cells more conservatively by allocating more cells compared to the number of required cells (the current traffic load). MSF [81] is the recent active Internet Draft (I-D) discussed by 6TiSCH working group of IETF. This scheduling function operates based on Orchestra [27] receiver based mode. If additive cell is needed, MSF uses 6P transactions for further cell allocation.

However, these methods have not yet been adequately validated on large

testbed. Moreover, though the distributed schedulers increased flexibility and scalability compared to the centralized schedulers, these methods still suffer from control traffic overhead required for updating the schedule.

Before having unicast active cells, each node should exchange packets by using broadcast cells. Generally, broadcast cells are shared by all the nodes in the networks and negotiating packets should be exchanged through these shared cells (e.g. minimal mode [67]). As a result, if routing topology changes frequently, slotframe schedule is updated frequently generating a lot of control packets on broadcast cells during the negotiation process. Since the number of broadcast cells are not enough to maintain low radio duty cycle, contention on broadcast cells will be increased and important packets might be lost decreasing network performance.

3.3 Autonomous Cell Scheduling.

In autonomous scheduling methods, each node schedules its own slotframe not generating any control traffic for slotframe schedule. It just uses only information already known.

Orchestra [27] 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 not generating any traffic overhead. This scheduling method uses three slotframes. One is used for TSCH EB exchange. Another is used for unicast packet exchange between RPL neighboring nodes. The other slotframe is used for broadcast packet exchange, which is also used as a default slotframe. Here, broadcast/default slotframe schedule is simple. It just uses minimal mode [67] schedule. Here, all the nodes in the network share one broadcast cell. Before unicast links are scheduled on

a unicast slotframe, they exchange RPL control packets by using this common shared broadcast cell. Whenever a node updates neighbor information regarding its RPL preferred parent and children, it also updates the unicast slotframe schedule by hashing node ID of corresponding neighbor. It proved its effectiveness and reliability through the evaluation using large testbeds. This scheduling method is used as the currently most commonly used cell scheduler. The source code of Orchestra is open and embedded on the currently most widely used operating system *Contiki* [43], which accelerates the widespread use of this scheduling method.

Though this novel scheduling method provides strong performance with simple strategy, some researchers have pointed out that Orchestra fails to provide reliable service on heavy traffic scenario [42][48].

Escalator [42] 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). As a result, the number of cells that should be scheduled on a slotframe is proportional to the number of nodes in its subtree. However, since nodes closer to the root have larger subtrees, these nodes inevitably suffer from long radio duty cycle. Moreover, it only focuses on upward traffic not allocating any unicast cell for downward traffic. Compared to Orchestra that hashes node ID of RPL neighboring nodes to calculate transmission cells or listen cells, Escalator hashes node ID of the message source to calculate cells and shifts the scheduled cells by the number of hop-count to achieve short latency end to end transmission.

e-TSCH-Orch [48] also tried to reduce latency of Orchestra focusing on collection (upstream) scenario as Escalator. Basically, e-TSCH-Orch follows the

same schedule of Orchestra. The only difference is as follows. When a node sends a packet to a neighbor node, it indicates the number of packets in its transmission queue on the packet footer. The message receiver receives the packet and schedules that amount of consecutive listen cells. By doing this, the message sender can quickly clean its transmission queue by transmitting remained packets in its transmission queue consecutively. 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 [79, 78].

Differently from the previous works focusing on the unicast transmission, Vallati *et al.* modified the broadcast slotframe of Orchestra [57]. 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.

3.4 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 au-

tonomous 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.

We also design and implement TRF-ALICE which is the extended version of ALICE considering the *real-time traffic load per each link*. With this extended method, each node can allocate multiple cells per slotframe for each directional link based on *the average number of transmission attempt per slot-*

frame occurred on each directional link. We prove that TRF-ALICE achieves more performance improvement compared to ALICE in terms of PDR, latency and radio duty-cycle in efficient way.

Chapter 4

Preliminary Study

To ground our study, we present a preliminary case study of Orchestra [27], 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.

4.1 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. Since this scheduling method uses already known information only to schedule the unicast slotframe, it is not required to have negotiation process in updating the slotframe schedule.

Specifically, each node needs only its MAC address (node ID), parent-child relationship (at the routing layer) and their MAC addresses for cell scheduling.

In their implementation, they set the last byte of MAC address as node ID, thereby, the possible node ID becomes the number between 0 to 255.

Orchestra provides three types of slotframes to deliver various traffic in LLN: (1) EB slotframe, (2) broadcast/default 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.

Three slotframes have their own slotframe schedule withdifferent slotframe length. They operate in parallel with different priority setting. Since lower-layer's connectivity is more important compared to the higher-layer's connectivity, EB slotframe has the highest priority. Before having unicast links scheduled on a unicast slotframe, all packets except TSCH EB are exchanged via boradcast/default slotframe. RPL control packets such as DIO are also exchanged via broadcast/default slotframe. To provide reliable network connectivity, this slotframe has higher prioity compared to unicast slotframe.

As a result, 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.

The EB slotframe is for exchanging TSCH 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. In Orchestra, each node sets its RPL parent node as its TSCH 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

$$t_o^{EB,Tx}(k) = Hash(\ ID(k)\ ,\ L_{SF}^{EB}\ ). \tag{4.1}$$

Note that node ID is *hashed*[1] (i.e., randomized) to mitigate EB collision. Based on Eq. (4.1), node k determines when to receive an EB from its TSCH time source, $t_o^{EB,Rx}(k)$, by calculating the EB transmission cell of its time source. This cell's time offset is the hashed value of the time source's node ID; Whenever a node changes its RPL preferred parent, it changes its TSCH time source to its new RPL preferred parent. As a result, $t_o^{EB,Rx}(k)$ 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 there is no unicast link scheduled on a unicast slotframe. Orchestra activates only one *fixed* cell, $(t_o, c_o) = (0, 1)$, in a broadcast slotframe as minimal mode [67] schedule. This cell is common shared broadcast cell. *All the nodes* in the network wake up simultaneously and listen at the corresponding channel. Only the the nodes who have the packets to transmit in the broadcast cell transmits the packet in this cell.

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. However, providing a large number of broadcast cells with short slotframe length might cause high radio duty cycle decreasing energy efficiency.

The unicast slotframe is for unicast 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*

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

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. (4.1), by hashing node ID as

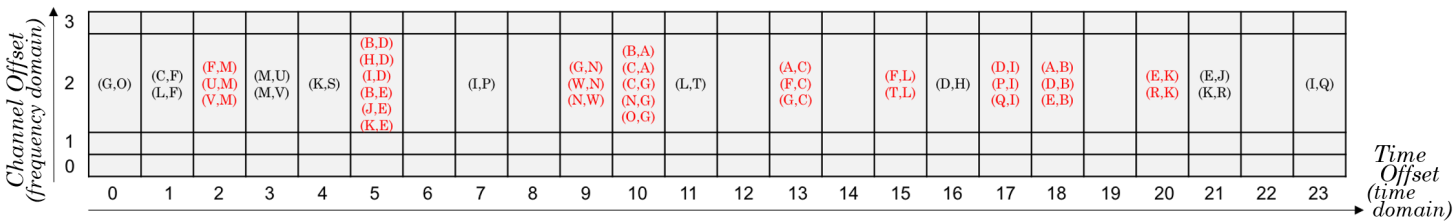
$$t_o^{UC,Rx}(k) = Hash(\ ID(k)\ ,\ L_{SF}^{UC}\ ) \tag{4.2}$$

where L_{SF}^{UC} is the unicast slotframe length.

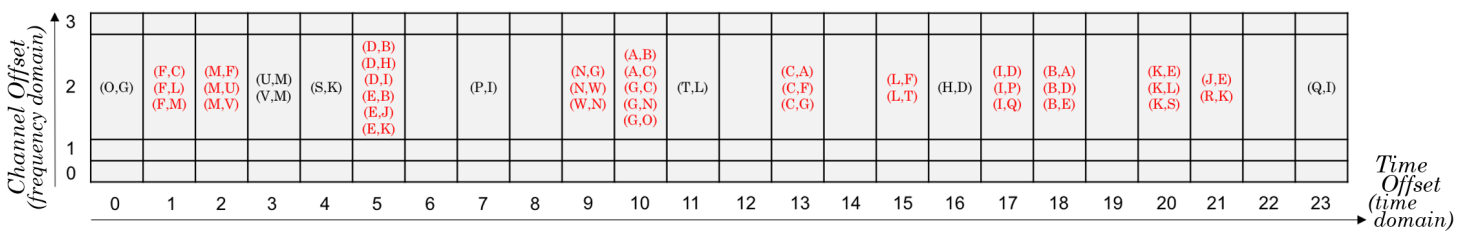
Node k's parent and children nodes extract $t_o^{UC,Rx}(k)$ from node k's ID and use the cell $(t_o, c_o) = (t_o^{UC,Rx}(k), 2)$ when sending a packet to node k. Since routing topology changes over time in general RPL settings, each node's neighbor relationship will be changed over time. Whenever each node changes its neighbor nodes, unicast transmission cells is changed reflecting its current neighbor relationship.

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. (4.2). 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 reflecting its current neighbor relationship.

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.



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



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

Figure 4.1 An example of Orchestra cell scheduling

An example of Orchestra cell scheduling with the 23-node topology in Figure 2.1(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.

4.2 Problems

We claim that node-based scheduling is inefficient. Figures 4.1(a) and 4.1(b) show scheduling examples of O-RB and O-SB, respectively, when 23 nodes are distributed/connected as in Figure 2.1(a). Note that the EB and broadcast slotframes, not shown in Figures 4.1(a) and 4.1(b), use channel offset 0 and 1 for active cells, respectively, never colliding with the active cells in the unicast slotframe.

The unicast slotframe in Figures 4.1(a) and 4.1(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 4.1(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 has only one transmission opportunity per unicast slotframe: node k needs N_k slotframes to transmit 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. Even though a node has a lot of packets in the transmission queue, it can transmit only one packet per a slotframe. As a result, its transmission queue will be

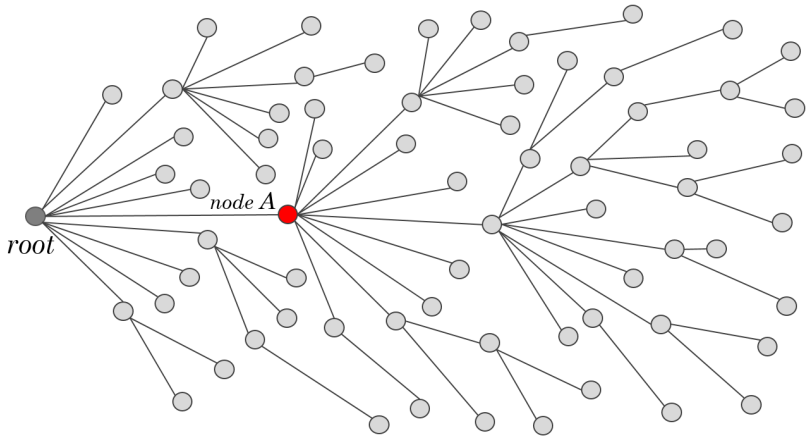


Figure 4.2 A snapshot of RPL routing topology

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

increased continuously and all the packets in its transmission queue will experience long queueing delay. Moreover, some incoming packets will be dropped due to full transmission queue. To give enough transmission opportunity, we can decrease the unicast slotframe length. However, it also increases radio duty cycle. Recall that 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 allocates N_k listen cells per a 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 [26][43] implementation) on the IoT-LAB [76] [77] public testbed located 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 4.2. We use MRHOF [30] for RPL objective function and 4 channels (15, 20, 25, 26) for TSCH ($L_{FHS} = L_{CH} = 4$).

As described on Section 2.2, enabling DAO-ACK option is important to establish reliable end-to-end downstream path. If DAO-ACK option is disabled, DAO will be just delivered in best-effort manner (Contiki's default setting disables DAO-ACK). With this reason, we enabled DAO-ACK option for all evaluation in this work.

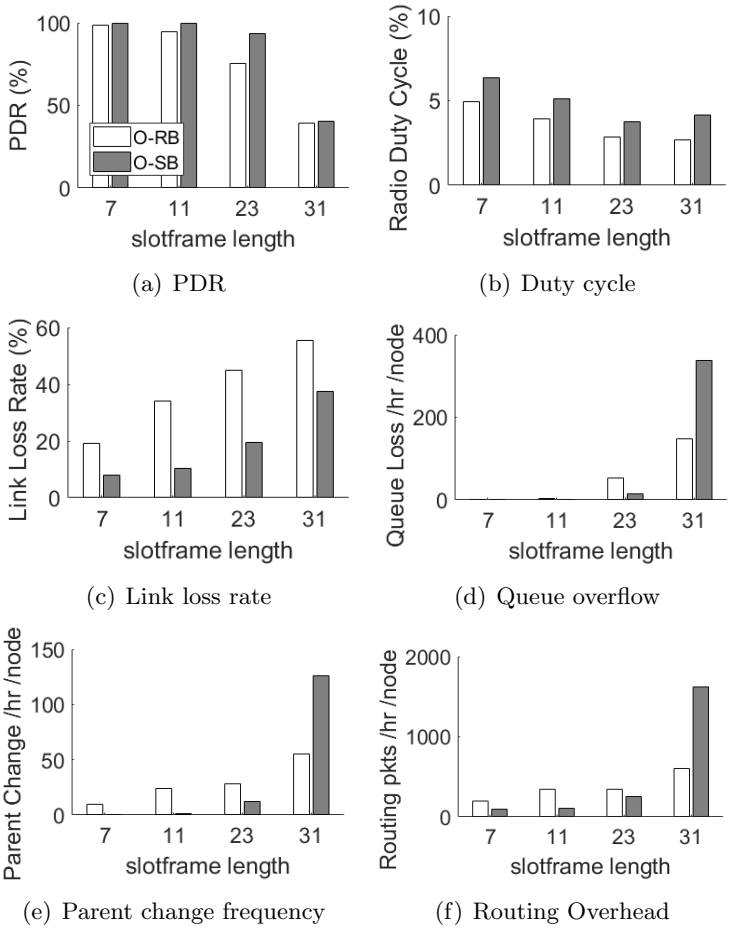
However, we could not use DAO-ACK option for O-RB since it increases the number of RPL control packets when the routing topology is unstable (in the scenario with high contention due to heavy traffic load or long slotframe length). When the routing topology is unstable, more RPL control packets are generated increasing contention on medium access. As a result, in case of O-RB, we found that the disadvantages of using DAO-ACK are greater than the benefits of using DAO-ACK. With this reason, we disabled DAO-ACK just for O-RB mode.

Instead, we used a large memory size and set the size of routing table and link level neighbor table to a number greater than the number of nodes used for evaluation. With this setting, end-to-end path was reliably established even without DAO-ACK in O-RB.

Moreover, in case of O-RB, each node transmits the packet at the receiver's listen cell. With this reason, even though its parent does not set it as its children, the packet will be successfully received by the receiver.

We generate heavy bidirectional traffic, 2 pkts/min of upward traffic from each node to the root node and 2 pkts/min downward traffic to each node from the root node. In total, the border router is required to deliver 268 pkts/min.

Figures 4.3(a) through 4.3(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



(a) PDR (b) Duty cycle

(c) Link loss rate (d) Queue overflow

(e) Parent change frequency (f) Routing Overhead

Figure 4.3 Performance of Orchestra

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

$L_{SF}^{BC} = 19$.

Trade-off about Slotframe Size. Firstly, we evaluated the effect of unicast slotframe length on performance of Orchestra. If one node has N_k neighbors, O-RB allocates one Rx cell and N_k Tx cells per a slotframe. Conversely, O-SB allocates one Tx cell and N_k Rx cells per a slotframe. Regardless of the used slotframe length, a node just allocates N_k+1 active cells per a slotframe. As a result, the longer the slotframe size, the more the number of idle timeslots.

With shorter slotframe length, the scheduled slotframe repeats more frequently with shorter period. As a result, a node has plenty more communication opportunity increasing packet delivery ratio. Instead, a node should more frequently wake up increasing radio duty cycle. Conversely, with longer slotframe length, the slotframe repeats slowly with longer period and a node can sleep more saving energy and showing low duty cycle. However, a node does not have enough communication opportunity resulting low packet delivery ratio.

Figures 4.3(a) and 4.3(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 ($\sim$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 4.3(c) and 4.3(d) show that both O-RB and O-SB suffer more link loss and queue loss as L_{SF}^{UC} increases, verifying 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 4.3(a) shows that Orchestra cannot provide reliable service with long slotframe length. In the figure, 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. Through a physical layer, a bit-level transmission is executed by transmitting radio signal. If physical layer provides unreliable service, a node cannot transmit a data packet to the receiver at a link layer. As such, reliability of link layer service directly affects the performance of routing layer. If a node pair fails to exchange data packet at a link layer, routing layer service cannot provide reliable end-to-end multihop packet forwarding service.

Figures 4.3(e) and 4.3(f) show that the routing layer is significantly affected by the performance degradation at the link layer. As L_{SF}^{UC} increases, both schemes suffer from more link losses and more queue losses. As a result, link level packet transmission between a node pair ends up with failure. Due to the link level transmission failures, expected transmission count (ETX) of each link increases. ETX is used as the main link quality metric in RPL. At a routing level, a node judge that the current path to the root is not good and changes the preffered parent. Even though it changes the preferred parent, the

same phenomenon repeats again continuously changing its preferred parent.

As described in the figures 4.3(e) and 4.3(f), 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 problem [36]. Figure 4.2 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 4.3(c) and 4.3(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 queueing delay and queue loss.

Figures 4.3(a) and 4.3(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. As described above, if one node has N_k neighbors, O-RB allocates one Rx cell and N_k Tx cells per a slotframe. Conversely, O-SB allocates one Tx cell and N_k Rx cells per a slotframe. In case of Rx cell, a node should listen unconditionally at the corresponding time and channel in case its neighbor transmits a packet. However, in case of Tx cell, a node wakes up only when it has any packet to transmit at the corresponding receiver at that cell. If there is no packet to

transmit, it just sleeps saving energy. As a result, O-SB shows higher radio duty cycle compared to O-RB when the same slotframe length is used.

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 4.3(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.

As described above, we enabled DAO-ACK option for O-SB. But disabled DAO-ACK option for O-RB due to severe contention at the Rx cells when there are a number of RPL control packets (when the topology is unstable continuously changing the preferred parent). As a result, when topology is unstable due to high link loss rate continuously changing the preferred parent, DAO and DAO-ACK messages are continuously generated overloading traffic on the network. Moreover, if DAO-ACK was not received, a node changes preferred parent until it receives DAO-ACK from the selected parent, which makes a node change the preferred parent more often in unstable topology. As a result, when the slotframe length is increased from 23 to 31, the number of parent change is extremely increased in case of O-SB compared to O-RB (Figure 4.3(e)) generating a lot of control packets at a routing level (Figure 4.3(e)). Due to a large number of routing control packets, its transmission queue is quickly filled with packets increasing queue loss (Figure 4.3(d)). Due to severely overloaded traffic on network, link loss rate is increased (Figure 4.3(c)). All these results ultimately lead to low PDR (Figure4.3(a)).

On the other hand, is it possible to escape from this trade-off: link loss (contention/collision) vs. queue loss (latency)? 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.

Chapter 5

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. Orchestra hashes node ID to make schedule table. As a result, by using hashed value, each node is directly mapped into a cell on a slotframe and this is used as Rx cell or Tx cell depending on the used mode. However, since each node has multiple directional links, Orchestra schedules multiple links as a group and allocates a group of links into a cell, which causes high link loss rate and high queue loss number. Recall that a transmission between a sender and a receiver occurs at a link layer. A link layer service should provide a method that can handle each directional link independently. Assigning multiple links as a group into a cell does not fit to a fundamental design philosophy of a link-layer service.

To resolve the problem, our intuition is that a unique cell should be allocated for *each directional link*, rather than a node. By doing this, we can handle each

directional link independently from other links. This section presents the design and implementation aspects of our proposed scheduling method, called *ALICE*.

5.1 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

$$t_o^{UC}(k,l) = Hash(\ \alpha ID(k) + ID(l)\ ,\ L_{SF}^{UC}\ ) \tag{5.1}$$

$$c_o^{UC}(k,l) = Hash(\ \alpha ID(k) + ID(l)\ ,\ L_{CH} - 1\ ) + 1. \tag{5.2}$$

Here the coefficient α is used to differentiate traffic directions,[1] e.g., link (k,l) vs. link (l,k). As in Eq. (5.2), 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

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

through RPL (DODAG topology), the number of directional links is $2N-2$. Given that 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.

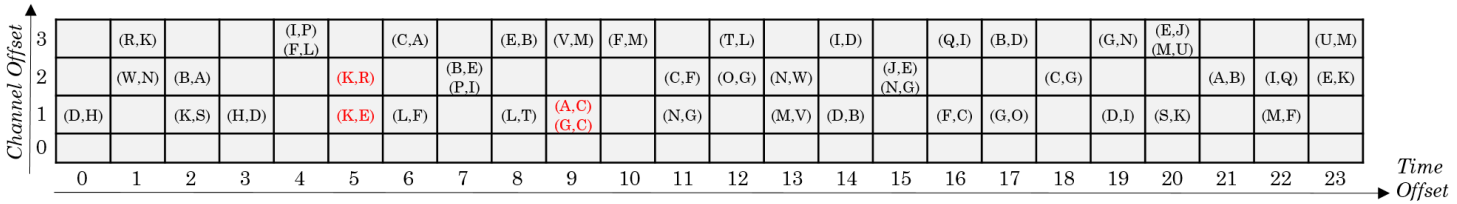
Since a hash function can return the same value for different links, ALICE may allocate one cell for multiple links. Then, some unlucky links scheduled at crowded cells will be disadvantaged continuously. To resolve the issue, ALICE changes cell schedules every unicast slotframe, which prevents specific 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

$$\begin{aligned} & t_o^{UC}(k, l, ASFN) \\ & = Hash(\ \alpha ID(k) + ID(l) + ASFN\ ,\ L_{SF}^{UC}\ ) \end{aligned} \tag{5.3}$$

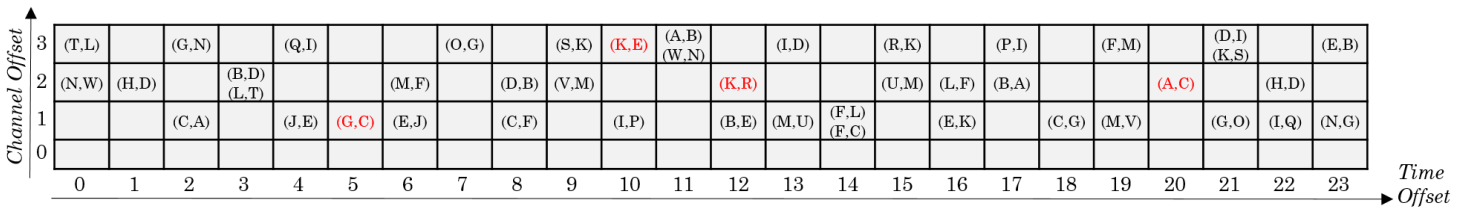
$$\begin{aligned} & c_o^{UC}(k, l, ASFN) \\ & = Hash(\ \alpha ID(k) + ID(l) + ASFN\ ,\ L_{CH} - 1\ ) + 1. \end{aligned} \tag{5.4}$$

Figure 5.1 shows an example of ALICE scheduling when 23 nodes are distributed/connected as in Figure 2.1(a). Figure 5.1(a) shows that ALICE utilizes multiple channels and allocates a unique cell for each directional link handling each directional link independently. Distributing directional links over two dimensional time-frequency schedule table, this method almost nullifies the contention, collision, and latency problems.

As exceptional cases, (K, R) and (K, E) are scheduled at the same timeslot



(a) ALICE schedule example at a unicast slotframe



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

Figure 5.1 An example of ALICE scheduling

An example of ALICE scheduling with the 23-node topology in Figure 2.1(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.

(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 5.1(b) shows the cell schedules at the next unicast slotframe, where all the previously overlapped links are distributed.

5.2 Design and Implementation

We implement ALICE on Contiki 3.0 and open the code.[2] 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 re-allocates cells by using Eqs. (5.3) and (5.4). If we use simple modulo operation, increasing the input value of the hash function by 1 increases the output value by 1. Then, the overlapped links at the same cell will be move into the other cell all together not resolving the problem (some unlucky nodes suffer from continuous disadvantage).

To avoid this unwanted situation, we use a combination of modulo and a simple 32-bit integer mix function [58] 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. As a result, increasing the input value of the hash func-

[2] [83] `https://github.com/skimskimskim/ALICE`

tion by 1 results in an unpredictable output value. At the same time, since all nodes use the same hash function and the same ASFN value, a pair of nodes connected by a link that should be scheduled in the same cell are scheduled in the same cell at the next slotframe iteration, thereby, they can communicate without problem even if the schedule is changed.

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 and transmitting at the corresponding cell can ruin the forwarding procedure. For example, even though a node enqueues a packet in its transmission queue at the current slotframe, the slotframe schedule will be changed at the next slotframe. As a result, it is meaningless to decide when and which cell to use to transmit packet at the time it enqueues the packet in the transmission queue. Instead, the transmission cell of each packet should be decided at the transmission cell.

To this end, 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. Recall that the routing topology might be changed over time. For example, a unicast packet can be enqueued to be sent to its link level neighbor at the current time and this packet is set to be transmitted at its receiver's listen cell. However, before transmitting this packet, the neighbor relationship has been changed.

Even if this packet is sent at a previously scheduled transmission cell, transmitting the packet at the previously scheduled cell is meaningless because the receiver of the packet is no longer its neighbor and no longer scheduled to listen at the cell scheduled in the past. 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 routing topology 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. When a node joins TSCH network, it initialize the network parameter such as ASN and make basic schedule table (line 1). After this, it sleeps and wakes up following the schedule table (line 2-14). Whenever a node wakes up, it is scheduled to do some operation: transmit a packet or listen at the current channel (line 3). In the current time slot, multiple operation can be scheduled. If Tx and Rx are scheduled at the same time, Tx operation has higher priority (line 4). A node selects one packet from the transmission queue by using `ALICE_Packet_Selector` (line 5). If there is a packet to transmit at this cell (line 6), it transmits the packet (line 7) and set the next active cell by using `ALICE_Next_Cell_Scheduler` (line 13). It sleeps until the next active cell (line 14). If a node has no packet to transmit at this Tx cell (line 6), it checks whether the current cell operation has Rx or not. If the current cell has Rx operation together with Tx operation (line 8), it listen at the corresponding channel (line 9). If the cell is scheduled for only Rx operation (line 10), it

Algorithm 1: ALICE Operation Manager

```
1  (t_o, c_o) ← 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) ← ALICE_next_cell_scheduler(t_o, ASFN);
14     sleep_until((t_o, c_o));
```

just listen at the corresponding channel (line 11). Every time it transmits and receives packet to and from their neighbor, RPL topology can be changed. If RPL topology is updated, it adds or removes corresponding cells updating the TSCH slotframe schedule (line 12). A node sets its next active cell (line 13) and sleeps until the next cell by using `ALICE_Next_Cell_Scheduler` (line 14). This operation is repeated at every active cell.

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 (line 1-2) until it finds the right packet (line 7-9) for the cell. If a packet's next hop node is not a RPL neighbor (line 3), the selector removes the packet from the queue ('Early Packet Drop') (line 4). As described earlier, this packet was enqueued in the packet queue previously, but the topology has been changed and the link layer destination (next hop) of this packet is not its RPL neighbor anymore. Even though it tries to transmit the packet, the receiver is no longer a neighbor and will not listen at the corresponding time and channel.

If the destination of the packet is still its neighbor (line 5), it checks whether the directional link from this node to the destination of the packet at the current $ASFN$ is scheduled on the input cell or not ('Packet-Cell Matching on the fly') (line 6-7). Here, it uses a function `ALICE_get_TX_cell` which outputs the cell of the directional link from the node to the *dest* at the $ASFN - th$ slotframe. If the output cell is same as the current cell (line 7), it outputs the packet and terminates the for loop (line9). If it finds no packet to transmit at the current cell, it outputs $NULL$ (line 10).

Algorithm 3 describes the `ALICE_Next_Cell_Scheduler`'s operation. Input value is the current time offset and the current $ASFN$ and the output is the

Algorithm 2: ALICE Packet Selector

input : cell (t_o, c_o), $ASFN$

output: packet

```
for pkt in PacketQueue do
    dest ← get_next_hop(pkt);
    if dest is not in RPL_NEIGHBOR_LIST then
        remove pkt from PacketQueue;
    else
        (t_o^pkt, c_o^pkt) ← ALICE_get_TX_cell(dest, ASFN);
        if (t_o^pkt, c_o^pkt) == (t_o, c_o) then
            packet ← pkt;
            terminate;
packet ← NULL;
```

Algorithm 3: ALICE Next Cell Scheduler

input : time offset t_o^{cur}, $ASFN$

output: next cell (t_o^{nxt}, c_o^{nxt})

$(t_o^{nxt}, c_o^{nxt}) \leftarrow NULL$;

for *cell* (t_o, c_o) *in Unicast_Slotframe_Active_Cell_List* **do**

  if $(t_o^{nxt}, c_o^{nxt}) == NULL$ *and* $t_o > t_o^{cur}$ **then**

    $(t_o^{nxt}, c_o^{nxt}) \leftarrow (t_o, c_o)$;

  else

    if $t_o > t_o^{cur}$ *and* $t_o < t_o^{nxt}$ **then**

      $(t_o^{nxt}, c_o^{nxt}) \leftarrow (t_o, c_o)$;

    else if $t_o == t_o^{nxt}$ **then**

      $(t_o^{nxt}, c_o^{nxt}) \leftarrow$ get_higher_priority_cell((t_o, c_o), (t_o^{nxt}, c_o^{nxt}));

if $(t_o^{nxt}, c_o^{nxt}) == NULL$ **then**

  ALICE_Unicast_SF_Scheduler($ASFN$+1) ;

  $(t_o^{nxt}, c_o^{nxt}) \leftarrow$ **ALICE_next_cell_scheduler**(-1, $ASFN$);

next active cell. The scheduler starts with initializing the next cell with $NULL$ value (line 1). When it finds a cell with larger time offset compared to the current time offset (line 3), it updates the next cell with this cell (line 4). After this, it finds the cell with time offset that is larger than the current time offset but smaller than the next time offset (line 6). If it finds this cell, it updates the next cell with this cell (line 7). If it finds the cell with time offset that is same as the current time offset (line 8), it selects the cell with higher priority (line 9). Regarding the priority, Tx operation has higher priority compared to the Rx operation.

After searching all the cells in the active cell list (line 2-9), the scheduler knows what is the next active cell. However, if the scheduler could not find the cell with a time offset that is larger than the current time offset (line 10), it means that the current cell was the last cell in the current $ASFN-th$ slotframe. Then, to prepare the next slotframe iteration, the scheduler reschedules the slotframe by increasing $ASFN$ by 1 ('Time Varying Scheduling') (line 11) and finds the first active cell by running `ALICE_Next_Cell_Scheduler` (line 12).

Chapter 6

Evaluation of ALICE

We evaluate the effectiveness of ALICE on the IoT-LAB [76] [77] with the same configuration as in Section 4.2. For comparison, we also evaluate both O-RB and O-SB of Orchestra.

6.1 Experimental Setting

We implemented ALICE on Contiki 3.0 and used the global public open testbed IoT-LAB located at Grenoble, France. We implemented Algorithm 1, Algorithm 2 and Algorithm 3 with equations 5.3 and 5.4. We used four 802.15.4 channels (15, 20, 25, 26) for TSCH with $L_{FHS} = L_{CH} = 4$. We used 68 M3 nodes[1] (MCU: *ARM Cortex M3, 32-bits, 72 Mhz, 64kB RAM*, Radio communication: *802.15.4 PHY standard, 2.4 Ghz*). One is used for both RPL border router (root) and TSCH PAN coordinator. The others are used for general nodes periodically

[1]Used node ID ranges from m3-208 to m3-286 at the Grenoble testbed.

transmitting packets to the root (upstream) and receiving packets from the root (downstream).

The duration of each TSCH timeslot is 10 ms. We used 397 for TSCH EB slotframe length and 19 for broadcast/default slotframe length. We increased the unicast slotframe length from 7 to 71. We varied the network density by changing the transmission power from -17dBm to 3dBm. We also changed the traffic load from 1 pkts/min to 6 pkts/min (both to and from the RPL root). We varied the traffic pattern by changing the upstream ratio from 10% (downstream dominant)to 90% (upstream dominant).

We use RPL storing mode. For RPL objective function, we used MRHOF [30] with ETX which is the Contiki's default RPL setting. The RPL topology changes over time and one snapshot of topology at -17 dBm transmission power is provided in Figure 4.2. At the initial time, RPL topology is evolving continuously changing the topology (low stability). For this reason, we waited 30 minutes before starting the experiment to achieve certain topology stability.

We enabled RPL DAO-ACK for both ALICE and O-SB to construct reliable end-to-end downstream path [16]. Though enabling DAO-ACK helps to establish reliable downstream path, additional overhead can be occurred due to 1) control packet overhead to send DAO-ACK and 2) path reestablishment (local repair) process that occurs when DAO-ACK is not received in time. In case of O-RB, due to insufficient Rx cell, enabling DAO-ACK resulted in high contention on Rx cell resulting dismal performance. With this reason, we disabled DAO-ACK just for O-RB.

6.2 Impact of Unicast Slotframe Size

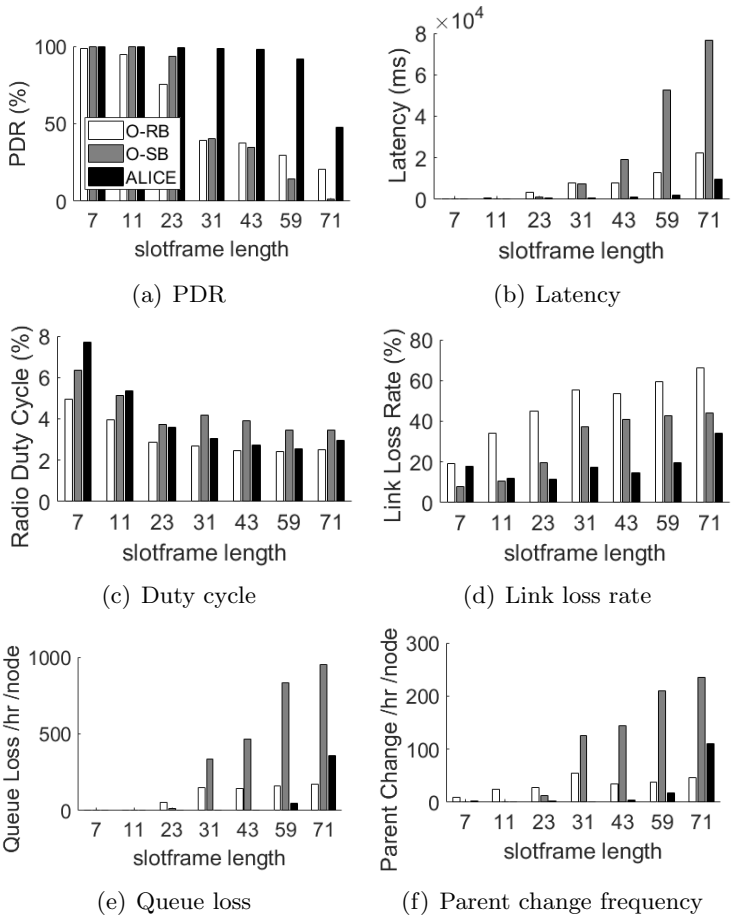
First, we evaluated the impact of unicast slotframe length. If we use short slotframe length, the slotframe frequently repeats over time giving enough communication opportunity. As a result, it results in high packet delivery ratio (PDR), low latency and high radio duty cycle. If we use long slotframe length, the slotframe slowly repeats over time giving low communication opportunity. As a result, it results in low PDR, high latency and low radio duty cycle.

In this experiment, we increased the unicast slotframe length from 7 to 71. We evaluated PDR, latency, radio duty cycle, link loss rate, queue loss number and parent change number. We used -17 dBm transmission power, thereby 68 nodes generate 6 hop network. Each node generates 2 upstream pkts/min to the RPL root and receives 2 downstream pkts/min from the root.

Figures 6.1(a) through 6.1(f) show various performance metrics of O-RB, O-SB, and ALICE according to the unicast slotframe size, when delivering bidirectional traffic.

Figures 6.1(a) and 6.1(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 4.2. 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. In case of O-SB, each node has only one Tx cell per a slotframe. As a result, even though a node has multiple packets in its transmission queue, it can transmit only one packet per slotframe length resulting long transmission queue. As a result, each packet should experience long queueing delay. Moreover, when the transmission queue is full, incoming packets are dropped. O-SB shows high queue loss number as described in Figure 6.1(e).

In case of O-RB, each node has only one Rx cell per each slotframe. As a



(a) PDR (b) Latency

(c) Duty cycle (d) Link loss rate

(e) Queue loss (f) Parent change frequency

Figure 6.1 Performance by the unicast slotframe length

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.

result, its Rx cell becomes crowded especially when the slotframe length is long causing high link loss rate as described in Figure 6.1(d). As slotframe length increases, the number of Rx cell per time decreases increasing link loss rate.

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 6.1(c) shows that ALICE obtains the above performance gain without sacrificing energy consumption. When a node has N neighbor, each node allocates 1 Rx cell and N Tx cells per slotframe in O-RB, 1 Tx cell and N Rx cells per slotframe in O-SB, and N Tx cells and N Rx cells per slotframe in ALICE, respectively. In case of Tx cell, a node does not have to wake up if there is no packet to send at the corresponding cell. However, in case of Rx cell, a node should always wake up at the scheduled cell to receive packet from its neighbor successfully. With this reason, allocating more Rx cells causes higher radio duty cycle. In terms of this, O-RB is the most energy efficient scheme since it allocates only one Rx cell per a slotframe.

O-SB and ALICE schedules Rx cells as much as the number of neighbor nodes resulting similar radio duty cycle. However, ALICE generally achieves lower radio duty cycle. In case of ALICE, it well distributes directional links over slotframe causing low link loss rate as described in Figure 6.1(d). However, since Orchestra schedules cells with node-based scheduling method, it handles multiple links as a group scheduling a group of links into a cell. As a result,

each cell is relatively crowded compared to ALICE resulting high link loss rate. Expected transmission count of each node is increased resulting high radio duty cycle.

When the slotframe length is short (e.g., when $L_{SF}^{UC} = 7$), ALICE shows high radio duty cycle with high link loss rate. ALICE uses 3 channel offsets for unicast slotframe while Orchestra uses 1 channel offset for unicast slotframe. As a result, when we use Orchestra, all the nodes wake up at the same channel with the same frequency hopping sequence. Even though multiple links are scheduled in a timeslot and one link was activated among them, all the nodes wake up at the same channel and there is no problem in communication with its corresponding neighbor. However, in case of ALICE with short slotframe length, multiple links are scheduled in a time slot and they are scheduled in the different channel offsets.

In a time slot, only one link that will be activated should be selected among the links. However, since each links are scheduled in the different channel offsets, their hopping sequence is different. As a result, even though one link was activated in the time slot, its corresponding neighbor might be wake up at the different channel. In this case, even though one message sender transmits the packet through the wireless channel, its corresponding receiver might be scheduled at the different channel for different operation, resulting collision. For this reason, ALICE shows high radio duty cycle when the slotframe length is short. However, the packet is retransmitted at the link layer achieving low link loss rate and high packet delivery ratio.

As L_{SF}^{UC} increases, O-RB and O-SB suffer significant link loss and queue loss, respectively, as shown in Figures 6.1(d) and 6.1(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 6.1(f)) and routing control 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.

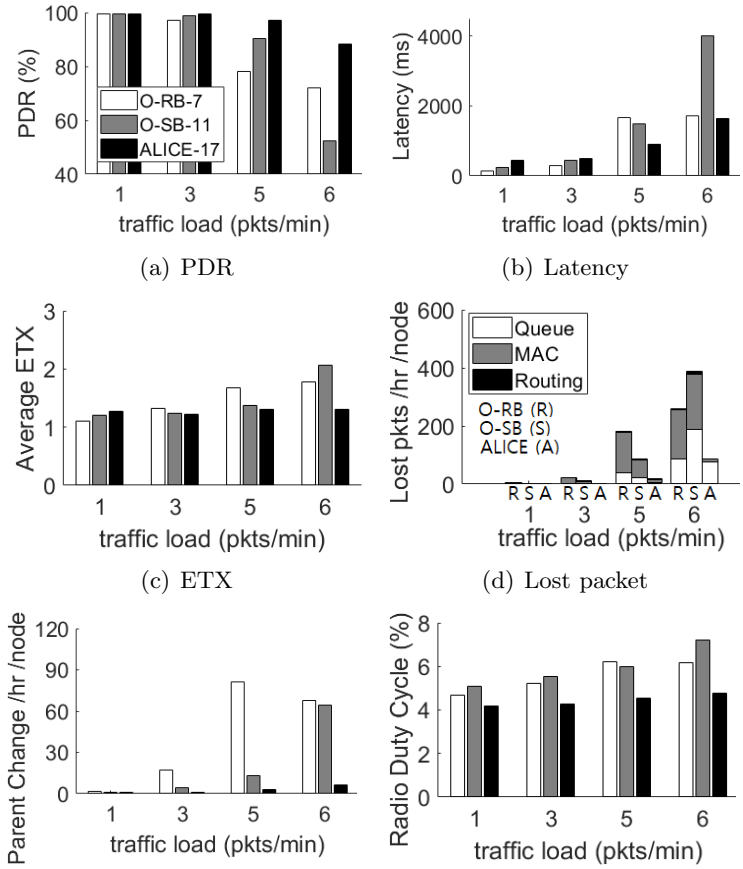
When allowing each scheme to use different L_{SF}^{UC} and targeting the high PDR (>99%), the best unicast slotframe sizes for O-RB, O-SB, and ALICE are 7, 11, and 23, respectively. When comparing these three cases, ALICE provides comparable PDR to O-RB and O-SB, while improving latency and duty cycle.

6.3 Impact of Traffic Load

Next, we investigate the impact of traffic load on each scheme's operation. Based on the results in Section 6.2, we use different slotframe length that best fits for each scheme. Thereby, we use the unicast slotframe size 7, 11, and 17 for O-RB, O-SB, and ALICE, respectively. Figures 6.2(a) through 6.2(f) show various performance metrics when traffic load varies from 1 pkts/min to 6 pkts/min.

Figure 6.2(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 even 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 6.2(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 the unicast slotframe length increases, ALICE incurs better latency



(a) PDR (b) Latency

(c) ETX (d) Lost packet

(e) Parent change frequency (f) Duty cycle

Figure 6.2 Performance by the traffic load

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

performance than the others.

Figure 6.2(c) shows that ALICE maintains low ETX (expected transmission count) under heavy traffic load, while both O-RB and O-SB incurs higher ETX. It means that the link error rate from a node to its parent in ALICE is relatively low compared to that of the other schemes.

Figure 6.2(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 6.2(e)). In the routing layer, lost packet is counted when 1) there is problem in routing header, 2) the packet is from unknown RPL instance, or 3) the packet was received from its parent (downward) but the receiver failed to find the next hop node (forwarding error).

AILCE shows extremely lower parent change number compared to the other schemes as described in Figure 6.2(e), which means that ALICE provide much higher routing stability. As a link-layer service, ALICE well distributes directional links over time and channel domain resulting low contention well-utilizing given resources. As a result, ALICE could provide solid and reliable link-layer service. Based on the robust link-layer service, RPL could construct more stable routing topology performing better on top of ALICE.

When the traffic load changes from 5 pkts/min to 6 pkts/min, performance of O-SB drastically decreases compared to that of O-RB (Figure 6.2(a)). The reason is, since O-SB is implemented with DAO-ACK enabled option, when the routing topology is not stable due to higher traffic overhead (6 pkts/min), a lot of RPL control packets were generated additionally changing the topology continuously increasing the parent change number (Figure 6.2(e)). As described above, RPL control packet overhead is generated due to 1) control packet overhead to send DAO-ACK and 2) path reestablishment (local repair) process that

occurs when DAO-ACK is not received in time. This creates negative cycle generating more RPL control packets again.

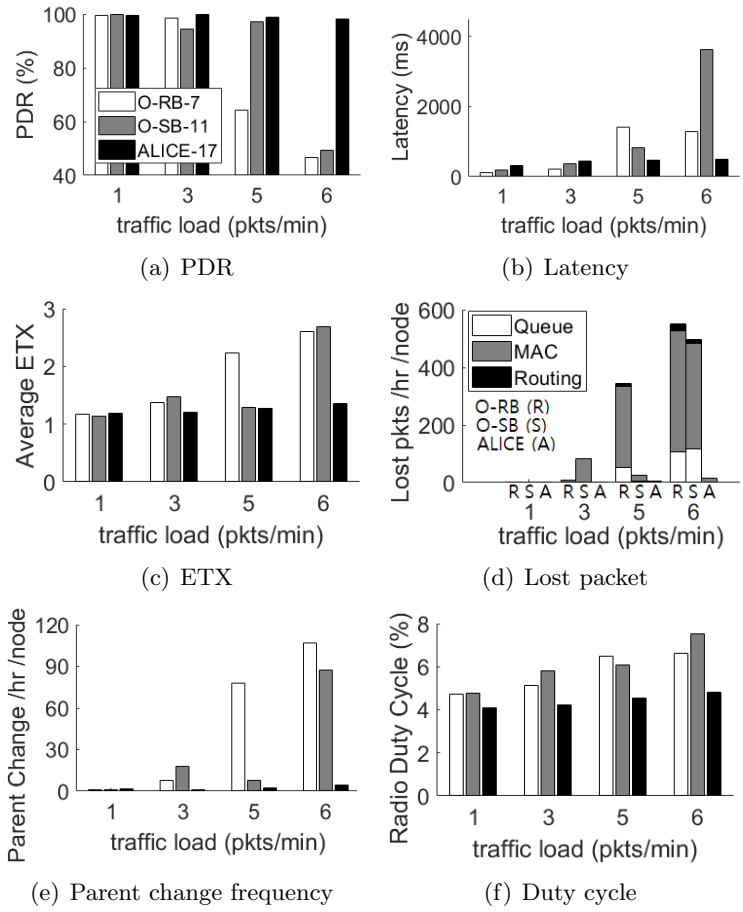
Due to RPL control packets, application level packets lost opportunity to be sent over the wireless medium causing low PDR (Figure 6.2(a)). Transmission queue became full of RPL control packets increasing latency (Figure 6.2(b)) and queueing drop (Figure 6.2(d)). Due to high link error rate and high lost packet number in the link layer (Figure 6.2(d)), its radio duty cycle is increased a lot (Figure 6.2(f)). In case of O-RB, it was implemented with DAO-ACK disabled option. As a result, it relatively less suffers from this phenomenon. Instead, it does not use DAO-ACK, thereby the DAO is just transmitted with best-effort manner creating unreliable end-to-end downstream path even in the scenario with low traffic load. In case of ALICE, even with DAO-ACK enabled option, it does support highgly reliable service in terms of both routing stability and link layer service.

Again, these results verify that ALICE addresses all the contention, collision, and latency problems efficiently. Lastly, Figure 6.2(f) shows that ALICE achieves better packet delivery performance even with lower duty cycle than the others; ALICE outperforms Orchestra in all aspects.

6.4 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. With 3 dBm transmission power, 68 nodes created shorter (4 hop) and denser routing topology. We perform the same experiments as in Section 6.3. Figure 6.3(a) through 6.3(f) show the results.

When comparing these results with those of Figures 6.2(a) through 6.2(f), it clearly shown that the performance of Orchestra worsens with high node



(a) PDR (b) Latency

(c) ETX (d) Lost packet

(e) Parent change frequency (f) Duty cycle

Figure 6.3 Performance by the traffic load (denser network)

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.

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*. In denser network, each node has more neighbors, thereby Rx cell is crowded scheduling multiple neighbor nodes to transmit in only one Rx cell per a slotframe. With this reason, O-RB suffers contention/collision even more resulting high link error rate and packet drops at the link-layer (Figure 6.3(d)). Similarly, O-SB allocates only one Tx cell for each node, where it sends packets 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 6.3(a)) due to severe link loss (Figure 6.3(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. In sparse network, each node has few neighbor. However, with denser network, each node has more neighbors with more links. Since ALICE schedules each directional links independently, ALICE schedules more communication opportunity per a slotframe in denser network, thereby ALICE could achieve significant performance improvement compared to that of Orchestra, even with denser topology.

Recall that the network topology became shorter with the increase of the transmission power to 3 dBm. With shorter network, end-to-end hop count is reduced. However, in case of Orchestra end-to-end latency was not improved (Figure 6.3(b)) due to high link loss rate. High link loss rate also resulted in

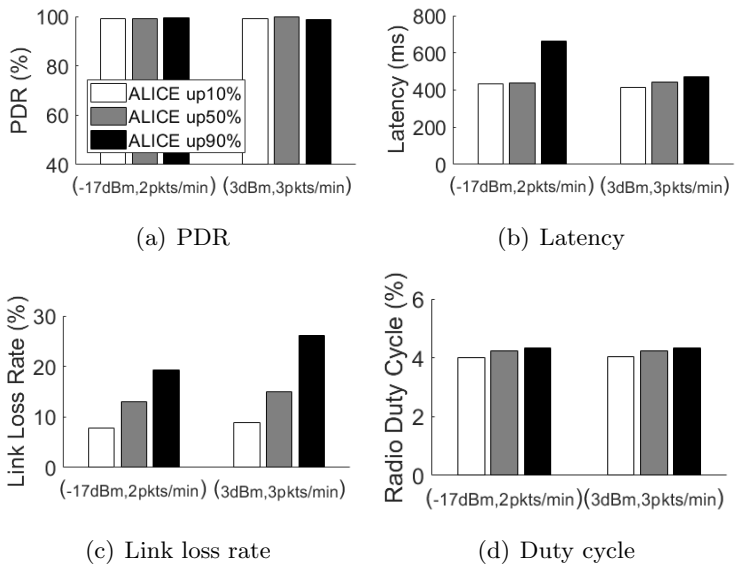
a large mount of lost packet in a link-layer (Figure 6.3(d)). However, in case of ALICE, it well distributed directional links over a slotframe well-utilizing given resources (channels and time). As a result, it could achieve low link loss rate (Figure 6.3(d)) and low end-to-end latency (Figure 6.3(b)) achieving high packet delivery performance (Figure 6.3(a)).

As shown in Figures 6.3(b) and 6.3(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.

6.5 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 (2 pkts/min at -17 dBm transmission power and 3 pkts/min at 3 dBm transmission power, respectively), 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 6.4(a) through 6.4(d). Here, we used 17 for ALICE unicast slotframe size as before.

Figure 6.4(a) shows that ALICE maintains high PDR regardless of traffic patterns, verifying its robustness. However, it does show somewhat performance degradation as upward traffic becomes dominant, longer latency (Figure 6.4(b)), more link losses (Figure 6.4(c)), and more radio duty cycle (Figure 6.4(d)). This is because ALICE allocates *only one cell* for every directional link. Although



(a) PDR

(b) Latency

(c) Link loss rate

(d) Duty cycle

Figure 6.4 Performance by the traffic pattern

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

this approach is better than Orchestra allocating more communication opportunity in efficient way, it still has a weakness when a specific link has to deliver much more traffic than other links. When the transmission queue is long due to heavy traffic load, if each packet's next-hop destination is different, multiple packets can be transmitted in a slotframe. However, if each packet's next-hop destination is one node, only one packet can be transmitted to the destination node.

When the upward packet is dominant, 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. As a result, in this case, even though ALICE schedules more links, only one specific link (upward link to the preferred parent) will be used dominantly. This again increases the transmission queue increasing end-to-end latency.

When the downward packets are dominant, downward packets are sent by a number of distributed downlink (links between each node and its children) cells resulting low link loss rate. When the upward packets are dominant, upward packets are sent by a few specific uplink (links between each node and its preferred parent) cells. As a result, compared to the downward packet dominant scenario, it causes higher link loss rate. As a result, each node should transmit each packet more increasing radio duty cycle.

Our results show that ALICE may need to use a smaller unicast slotframe to support upward-focused traffic than downward-focused traffic to give enough communication opportunity between a node and its preferred parent. 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.*

Chapter 7

TRF-ALICE: "Traffic Load"-based Cell Scheduling

Though ALICE achieves high throughput with low energy consumption, it still has limitation in that each directional link can have only one transmission or listen opportunity per a slotframe. On RPL network which is based on DODAG structure, a node close to the root in the tree has many packets to forward between the root and the nodes in its sub-network. To handle this heavy traffic load in efficient way, a node with more traffic load should have more communication opportunity considering its real-time traffic load compared to the nodes with low traffic load (e.g. leaf nodes). However, until now, there is no *autonomous* TSCH cell scheduling method that is reflecting the real-time traffic load. There are only traffic load -based schedulers requiring a lot of communication overhead to exchange information on real-time traffic load between neighboring nodes [44][45][49][52][63][72].

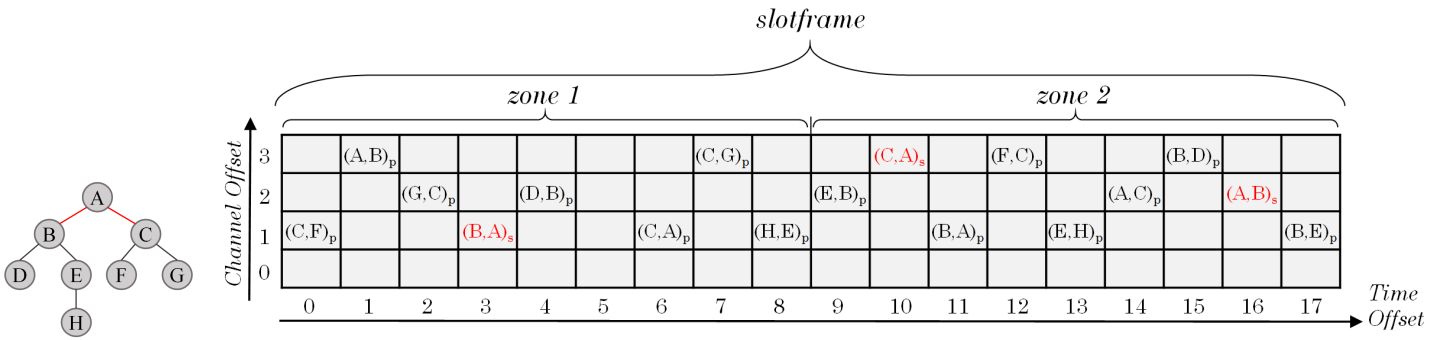
Here, we propose a novel and *autonomous* TSCH cell scheduling method

that is reflecting the *real-time traffic load*. This scheduling method is based on ALICE's unique *directional link -based* scheduling method we previously introduced. We call this extended version of ALICE as TRF-ALICE.

7.1 Overview

A basic scheduling rules of TRF-ALICE follow key idea of ALICE: it autonomously schedules each directional link on a slotframe by using hash function and handles each directional link independently. The only one difference is, a slotframe is divided into N_z different regions. Based on the traffic load, a node can use only one region or multiple regions in a slotframe. Each directional link should schedule at least one cell and the cell is scheduled at the primary zone. We call this default cell as a primary cell. Based on the link ID, each directional link chooses one primary zone among N_z zones in a slotframe and the primary cell will be scheduled on this primary zone. For the selection of primary zone, time information ($ASFN$) is used together, thereby the primary zone of each link changes over time. In the selected zone, a directional link is scheduled following ALICE's hash-based scheduling method. If the traffic load increases on that link, a node allocates more cells based on the current traffic load and the link quality. Multiple cells for that link are scheduled on different zones. By doing this, a node can allocate different number of cells for each directional link based on the real-time traffic load and the link quality.

Figure 7.1 (b) shows a scheduling example of TRF-ALICE of 8-node RPL topology (figure 7.1(a)). In this example, unicast slotframe length L_{SF}^{UC} is 18 and the number of zones N_z is two. Each directional link schedules at least one cell for the link. We call this default cell as a *primary* cell. Based on hash function,



(a) RPL topology

(b) TRF-ALICE schedule example when the number of zones (N_z) is two

Figure 7.1 An example of TRF-ALICE scheduling

An example of TRF-ALICE scheduling with 8-node topology (a) and the unicast slotframe length $L_{SF}^{UC} = 18$ when the number of zones N_z is two (b). A slotframe is divided into N_z different zones. Based on hash function, a cell for each directional link is scheduled on one of N_z zones. Each directional link can have up to N_z cells and each cell is scheduled in the different zones. Each directional link should schedule at least one cell and this default cell will be used as a primary cell. Additionally allocated cells are used only when they are needed based on the current traffic load. In this example (b), any directional link from X to Y is scheduled on one primary cell which is indicated as $(X,Y)_p$ and marked with black text. Depending on the current traffic load, some directional link from X to Y can have additional (secondary) cell which is indicated as $(X,Y)_s$. In this example, 3 links have secondary cells and they are marked with red text.

the primary cell for each directional link is scheduled on one of multiple zones. If additional cell is needed for a directional link because of the increased traffic load or bad link quality, a node can schedule multiple cells for that link at the zones not used by the primary cell. The additional cells are used only when they are needed. In the scheduling example (figure 7.1(b)), any directional link from node X to node Y schedules its primary cell which is indicated as $(X, Y)_p$ and marked with black text. Depending on the real-time traffic load, some directional links can schedule additional (secondary) cells. In the figure 7.1(b), additionally allocated cell for the link (X, Y) is indicated as $(X, Y)_s$. In this example, 8 nodes have 14 directional links and their primary cells are scheduled on a slotframe. Node A, B and C have more traffic load than the others and they schedule additional cells ($(A, B)_s$, $(B, A)_s$ and $(C, A)_s$) to support heavy traffic load. In the figure 7.1(b), these additionally allocated cells are marked with red text.

Primary zone is decided by the hashed value of link ID and time information ($ASFN$). Thereby, each node uses different zone for its primary cell allocation and the location of the primary zone is also changes over time (*time-varying* scheduling). Since node B and C in the figure 7.1(a) have more packets to forward between the root and their sub-tree nodes, they need more cells compared to other nodes. In this case, additional cells can be allocated to support that heavy traffic load. In this example (figure 7.1(b)), the number of zones N_z used in a slotframe is two. The maximum number of cells that each directional link can have is $N_z = 2$ and each cell for a link should be scheduled on different zones on a slotframe. In TRF-ALICE, the most interesting point is that each node decides the number of cells required for each link in *autonomous* way not requiring any negotiation process between the transmitter and the receiver of

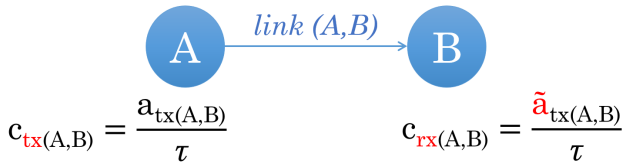


Figure 7.2 Concept of the number of cells decision in TRF-ALICE

Example of deciding the number of cells for the link (A, B). There is a directional link from node A to node B. A node A is a transmitter and a node B is a receiver. Node A decides the number of transmission (tx) cells for the link (A, B) based on its average number of transmission attempt per a slotframe. Node B decides the number of listen (rx) cells by estimating the average number of transmission attempt per slotframe of node A (transmitter). Considering the target cell utilization (τ), more number of cells are allocated than the number of transmission attempt.

a directional link.

Then, how each node *autonomously* decides the number of cells to allocate for each directional link? The answer is simple. A transmitter decides the number of tx cells for each link based on *the number of transmission attempt per slotframe* and a receiver decides the number of rx cells for each link based on the *estimated* number of transmitter's transmission attempt per slotframe.

7.2 Design and Implementation

Figure 7.2 illustrates the concept of how TRF-ALICE *autonomously* decides the number of cells to allocate for each directional link. In the figure, there is a directional link from node A to node B. Here, a node A is a transmitter and a node B is a receiver. Two nodes decide the number of tx/rx cells for

the link (A, B) autonomously without exchanging any information during their cell scheduling operation. The number of tx cells scheduled by node A for the link (A, B) is $c_{tx(A,B)}$. The number of rx cells scheduled by node B for the link (A, B) is $c_{rx(A,B)}$. Here, the *target cell utilization* τ is also considered. $a_{tx(A,B)}$ is the number of transmission attempt of node A to transmit packet to node B per a slotframe. $\tilde{a}_{tx(A,B)}$ is a receiver's *estimated* value of transmitter's $a_{tx(A,B)}$. To estimate $\tilde{a}_{tx(A,B)}$ at the receiver side, the receiver observes the result of listen operation on each listen (rx) cell. The result of listen operation can be categorized into four: (1) Channel is idle (silent). (2) Successfully received packet from node B (transmitter) and sent acknowledgement back to node B. (3) Seen packet but it is for the other node pairs. (4) Transmission energy detected but cannot parse the packet (collision detected).

Let's start with a transmitter's cell allocation for the link (A, B). When a transmitter transmits a packet, it results in success or collision. When the transmitter receives link-layer acknowledgement from the corresponding receiver, it means that the packet was successfully received by the receiver. When the transmitter did not received link-layer acknowledgement from the receiver, it means that the packet was not successfully sent to the receiver. $\hat{a}_{tx(A,B)}$ is the number of transmission attempt of node A to node B during the current slotframe. $\hat{a}^{success}_{tx(A,B)}$ and $\hat{a}^{fail}_{tx(A,B)}$ are the number of successful transmission and failed transmission of node A to node B during the current slotframe, respectively. Then we obtain the following equation.

$$\hat{a}_{tx(A,B)} = \hat{a}^{success}_{tx(A,B)} + \hat{a}^{fail}_{tx(A,B)} \tag{7.1}$$

As illustrated in figure 7.2, for the decision of the number of transmission cells $c_{tx(A,B)}$, the historical value of transmission attempt is needed. To obtain the historical values, we use exponentially weighted moving average (EWMA). In our implementation, we use EWMA α_{EWMA} from 0.05 to 0.13 depending on the used unicast slotframe length. Using EWMA, at the end of each slotframe, we update $a^{success}_{tx(A,B)}$ and $a_{tx(A,B)}$. $a^{success}_{tx(A,B)}$ and $a_{tx(A,B)}$ are the historical values of the average number of *successful* transmission per a slotframe and the average number of *total* transmission attempt per a slotframe, respectively. Then, the number of tx cells required for the link (A,B) is easily obtained with the following equation where τ is the *target cell utilization*. In our implementation, we use $\tau = 0.75$.

$$c_{tx(A,B)} = \frac{a_{tx(A,B)}}{\tau} \tag{7.2}$$

When updating the historical value of the transmission attempt per slotframe $a_{tx(A,B)}$, the value is strictly limited by the value $2 \cdot a^{success}_{tx(A,B)}$ to avoid the case where the transmitter allocates more number of cells than the number of listen cells the receiver allocates. When the $a_{tx(A,B)}$ exceeds $2 \cdot a^{success}_{tx(A,B)}$, it means that the collision probability of the link exceeds 50%. This means that the receiver might not be scheduled to listen when the transmitter transmits a packet or the channel is too bad to be used for communication. To avoid useless cell allocation, when updating $a_{tx(A,B)}$, the value is strictly limited by $2 \cdot a^{success}_{tx(A,B)}$.

Basically, with TRF-ALICE, each node sets the number of tx cells or rx cells for each directional link for the next slotframe at the end of the current

slotframe. After the decision of the number of tx cells or rx cells for each directional link, next slotframe schedule is conducted considering time information (*time-varying* scheduling).

Then, let's see how the receiver decides the number of rx cells for the link (A, B). When the receiver listens the channel at each rx cell, listen operation results in one of four categories: (1) Channel is idle (silent). (2) Successfully received packet from node B and sent acknowledgement back to node B. (3) Seen packet but it is for the other node pairs. (4) Transmission energy detected but cannot parse the packet (collision detected). When the receiver listens the channel for the link (A, B), it counts the number of each results. $\hat{r}^{idle}_{rx(A,B)}$, $\hat{r}^{success}_{rx(A,B)}$, $\hat{r}^{others}_{rx(A,B)}$ and $\hat{r}^{collision}_{rx(A,B)}$ represents (1) the number of rx slots that was idle, (2) the number of rx slots that received packet successfully from the node A (transmitter), (3) the number of rx slots where packet seen but it is for the other node pairs and (4) the number of rx slots where collision was detected *during the current slotframe*, respectively. Here, the total number of rx slots activated at the current slotframe is $\hat{r}_{rx(A,B)}$. Then, we obtain the following equation regarding the current slotframe.

$$\hat{r}_{rx(A,B)} = \hat{r}^{idle}_{rx(A,B)} + \hat{r}^{success}_{rx(A,B)} + \hat{r}^{others}_{rx(A,B)} + \hat{r}^{collision}_{rx(A,B)} \tag{7.3}$$

Recall that the transmission operation has higher priority compared to the listen operation. Moreover, in hash-based autonomous scheduling method, multiple cells might be scheduled on a timeslot. In this case, a node can be activated at only one channel offset since a node has only one radio interface and a cell with higher priority is selected. As a result, some listen cells might not be ac-

tivated. To reflect this case, we also consider the number of inactivated rx cells and it can be easily counted since the receiver knows the number of cells it allocated for the link (A, B) at the current slotframe ($\hat{c}_{rx(A,B)}$). Then, the number of inactivated rx cells can be calculated as follows:

$$\hat{r}^{inactivated}_{rx(A,B)} = \hat{c}_{rx(A,B)} - \hat{r}_{rx(A,B)} \tag{7.4}$$

By using EWMA, the receiver manages the historical value $\tilde{a}_{tx(A,B)}$ which is the estimated transmission attempt of node A (transmitter). At the end of each slotframe, it is updated by following equation:

$$\begin{aligned} &\tilde{a}_{tx(A,B)} = (1 - \alpha_{EWMA}) \cdot \tilde{a}_{tx(A,B)} \\ &+\alpha_{EWMA} \cdot (\hat{r}^{success}_{rx(A,B)} + \tilde{p}^{tx}_{tx(A,B)} \cdot (\hat{r}^{collision}_{rx(A,B)} + \hat{r}^{inactivated}_{rx(A,B)})) \end{aligned} \tag{7.5}$$

When the result of listen operation is success, it means the transmitter A transmitted packet. So the receiver increases the estimated value of transmitter's transmission attempt during the current slotframe by 1. If there was no packet seen (idle rx slot) or there was packet seen but it was for other pair of nodes, it means that the transmitter A did not transmitted the packet to the receiver B. Thereby, the receiver does not increase the estimated value of transmitter's transmission attempt during the current slotframe. In case of collision, the receiver does not know whether the transmitter was involved in the collision event or not. However, the receiver knows the estimated value of the transmitter's average transmission attempt per a slotframe ($\tilde{a}_{tx(A,B)}$) and the number of rx slots allocated for the current slotframe ($\hat{c}_{rx(A,B)}$). Then, the re-

ceiver can estimate the transmitter's transmission probability at the scheduled cell that is :

$$\tilde{p}^{tx}_{tx(A,B)} = \frac{\tilde{a}_{tx(A,B)}}{\hat{c}_{rx(A,B)}} \tag{7.6}$$

Similarly, in case of inactivated rx slot, the receiver does not know whether the transmitter transmitted the packet or not. So it assumes that the transmitter transmitted the packet at the scheduled cell with the probability $\tilde{p}^{tx}_{tx(A,B)}$.

Then, by considering the target cell utilization τ, the number of rx cells required for the next slotframe is obtained by the following equation:

$$c_{rx(A,B)} = \frac{\tilde{a}_{tx(A,B)}}{\tau} \tag{7.7}$$

Based on the equations (7.2) and (7.7), each node decides the number of tx or rx cells for each directional link. To avoid (1) frequent oscillation between different number of cells and (2) the case where the transmitter allocates more cells than the the number of rx cells the corresponding receiver allocated, we use increase thresholds and decrease thresholds.

A threshold $th^{X\to Y}_{tx}$ indicates the transmitter's the number of tx cell update threshold from the number X to Y. If Y is larger than X and the current tx cell allocation number $\hat{c}_{tx(A,B)}$ is X, the transmitter increases the tx cell allocation number from X to Y for the next slotframe if $c_{tx(A,B)}$ is larger than $th^{X\to Y}_{tx}$. If X is larger than Y, the transmitter decreases the tx cell allocation number from X to Y for the next slotframe schedule, if $c_{tx(A,B)}$ is smaller than $th^{X\to Y}_{tx}$.

In the similar way, the receiver also has receiver's thresholds ($th_{rx}^{X \to Y}$). We set the receiver's threshold $th_{rx}^{X \to Y}$ lower than that of the transmitter's to avoid the case where the transmitter's cell allocation number is larger than that of the receiver's allocation.

In our implementation, we set the transmitter's thresholds as $th_{tx}^{1 \to 2}$ =1.0, $th_{tx}^{2 \to 1}$ =0.9, $th_{tx}^{2 \to 4}$ =2.0 and $th_{tx}^{4 \to 2}$ =1.85 and the receiver's thresholds as $th_{rx}^{1 \to 2}$ =0.85, $th_{rx}^{2 \to 1}$ =0.75, $th_{rx}^{2 \to 4}$ =1.8 and $th_{rx}^{4 \to 2}$ =1.6.

Each node *autonomously* decides the number of cells to allocate for each directional link for the next slotframe schedule. For cell scheduling, we use different cell ID (c_{id}) for each cell scheduled for a directional link. c_{id} of the primary cell is 0 and additionally allocated cells have different c_{id} in increasing order. Decision of channel offset is same as equation (5.4). Time offset of the link (k, l) is scheduled as follows.

When the number of zones in a slotframe is N_z, the maximum number of cells available for a directional link is N_z. Depending on the location of the primary zone at the $ASFN^{th}$ slotframe, the location of additionally allocated cells are decided. The location of the primary zone for the link (k, l) at the $ASFN^{th}$ slotframe is obtained as follows:

$$z_{pr}(k, l, ASFN) = Hash(\alpha ID(k) + ID(l) + ASFN, N_z). \qquad (7.8)$$

Then, $z_{pr}(k, l, ASFN)$ can have value from 0 to N_z-1. Based on the c_{id} of the cell, the cell is scheduled on the primary zone or the other zones. The zone selection equation is simple as follows:

$$z(k, l, ASFN, c_{id}) = mod(z_{pr}(k, l, ASFN) + SHIFT[c_{id}], N_z). \quad (7.9)$$

Here, we use the array $SHIFT$ to allocate cells efficiently in terms of queueing delay. When we use N_z=2, we use $SHIFT = [0, 1]$. When we use N_z=4, we set $SHIFT = [0, 2, 1, 3]$. As a result, from the primary zone selected, additionally allocated cell shifts the zone by $SHIFT[c_{id}]$.

Finally, time offset of a cell with cell ID c_{id} of a link (k, l) at the $ASFN^{th}$ slotframe is decided by the following equation:

$$\begin{aligned} t_o^{UC}(k, l, ASFN, c_{id}) \ &= \ z(k, l, ASFN, c_{id}) \cdot L_{SF}^{UC}/N_z \\ &+ \ Hash(\ \alpha ID(k) + ID(l) + ASFN \ , \ L_{SF}^{UC}/N_z). \end{aligned} \quad (7.10)$$

We implemented TRF-ALICE on Contiki 3.0. The implementation of TRF-ALICE is simple. We just extended ALICE to TRF-ALICE by implementing following functions: (1) Each node counts the number of successful transmission and failed transmission for each directional tx links at the end of tx slot. It also counts the number of activated rx slots and their results (idle, successful reception, packet for others, collision) for each directional rx links at the end of rx slot. (2) At the end of the slotframe, each node updates EWMA averaged the number of transmission attempt and successful transmission attempt per slotframe for tx links and the estimated number of transmission attempt per slotframe for rx links. (3) Finally, a node decides the number of tx/rx cells needed for each directional link for the next slotframe and schedules the next

slotframe by using equations (7.10) and (5.4).

Chapter 8

Evaluation of TRF-ALICE

We evaluate the effectiveness of TRF-ALICE on the IoT-LAB [76][77] with the similar configuration as in Section 6. Since ALICE outperforms in all aspects compared to Orchestra, we just compare the performance of TRF-ALICE with ALICE.

8.1 Experimental Setting

We implemented TRF-ALICE on Contiki 3.0 and used the global public open testbed IoT-LAB located at Grenoble, France. We used four 802.15.4 channels (15, 20, 25, 26) for TSCH with $L_{FHS} = L_{CH} = 4$. We used 62 M3 nodes[1] (MCU: *ARM Cortex M3, 32-bits, 72 Mhz, 64kB RAM*, Radio communication: *802.15.4 PHY standard, 2.4 Ghz*). One is used for both RPL border router (root) and TSCH PAN coordinator. The others are used for general nodes periodically

[1]Used node ID ranges from m3-290 to m3-358 at the Grenoble testbed.

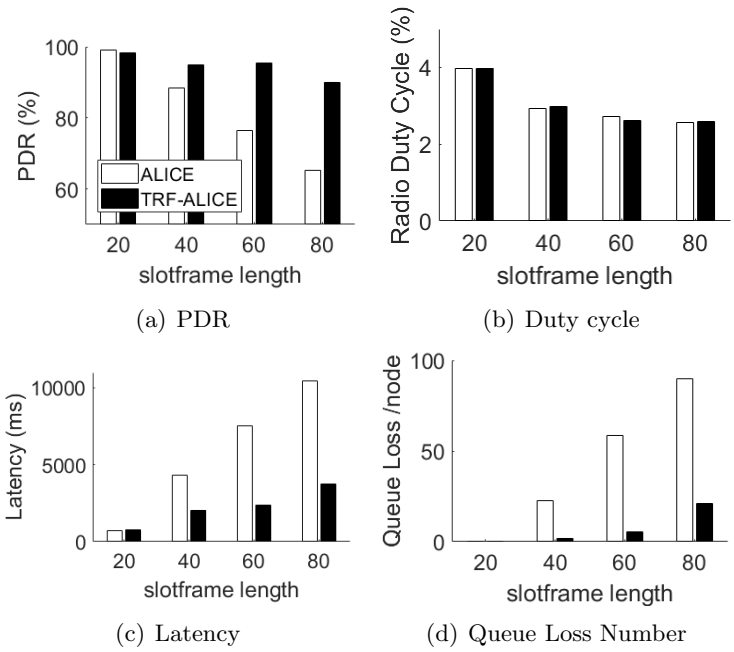
transmitting packets to the root (upstream) and receiving packets from the root (downstream).

The duration of each TSCH timeslot is 10 ms. We used 397 for TSCH EB slotframe length and 17 for broadcast/default slotframe length. We increased the unicast slotframe length from 20 to 80. We used the transmission power of -17dBm. We changed the traffic load from 4 pkts/min to 15 pkts/min (both to and from the RPL root).

8.2 Impact of Unicast Slotframe Size

First, we evaluated the impact of unicast slotframe length. If we use short slotframe length, the slotframe frequently repeats over time giving enough communication opportunity. As a result, it results in high packet delivery ratio (PDR), low latency and high radio duty cycle. If we use long slotframe length, the slotframe slowly repeats over time giving low communication opportunity. As a result, it results in low PDR, high latency and low radio duty cycle.

Figures 8.1(a) through 8.1(d) show various performance metrics of ALICE and TRF-ALICE ($N_z = 4$) according to the unicast slotframe size, when delivering bidirectional traffic. Since the nodes closer to the root have many packets to forward between the root and their sub-tree nodes, it needs more communication opportunity compared to other nodes. In case of ALICE, each node has communication opportunity as large as the number of directional links it is involved. As a result, if the number of neighbor is same, a node has same communication opportunity regardless of its location of the node in the tree. In case of TRF-ALICE, each node observes timely traffic load per each directional link and allocates additive cells if they are needed. As a result, even with a long slotframe length, TRF-ALICE could achieve high PDR (figure 8.1(a)).



(a) PDR (b) Duty cycle

(c) Latency (d) Queue Loss Number

Figure 8.1 Performance by the slotframe length

Performance of ALICE and TRF-ALICE ($N_z = 4$) according to unicast slotframe size under bidirectional traffic (4 pkts/min from each node and 4 pkts/min to each node). TRF-ALICE provides high PDR and energy efficiency even with large slotframe size.

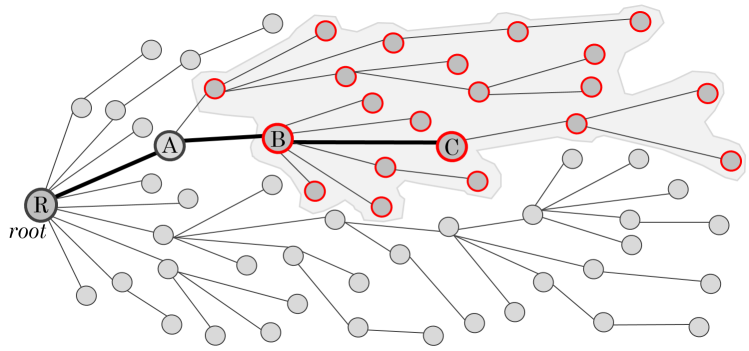
From the log, we found that only a few links of a few nodes with a large subtree utilized additional cells. As a result, radio duty cycle increases only slightly (figure 8.1(b)). TRF-ALICE also reduced end to end latency (figure 8.1(c)) while supporting high PDR. Since TRF-ALICE provides more transmission opportunity for the directional links with high traffic load, the links with heavy traffic load could clean their queue more quickly by utilizing multiple cells in a slotframe. As a result, its queue could be kept short showing low queue loss number (figure 8.1(d)). It means that each packet stays in the queue in short time reducing queueing delay. Thereby, TRF-ALICE could achieve high PDR, low latency, low queue loss number showing a similar amount of radio duty cycle compared to ALICE.

8.3 Impact of dynamic traffic load

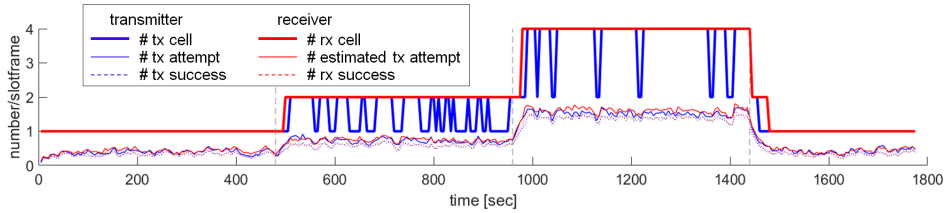
We evaluated the number of cells allocation of TRF-ALICE by time changing the traffic load over time. Here, we used the fixed RPL topology to control the traffic load for specific links. The used unicast slotframe length is 40 and N_z=4. After the 10 minutes initialization time, the experiment lasts for 30 minutes.

Figure 8.2(a) shows the used RPL topology. We used 62 nodes. During the 30 minutes experiment, the root generates 2 pkts/min to each node of 61 nodes. Receiving 2 pkts/min from the server, each node generates 2 pkts/min to the server.

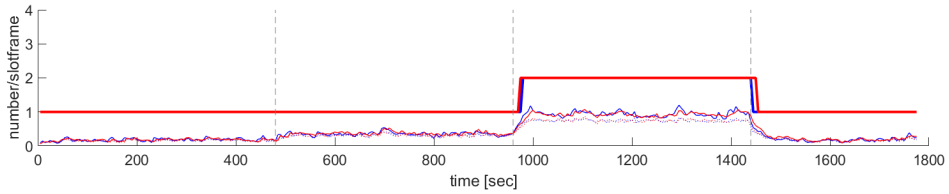
In the figure 8.2(a), 21 nodes marked with red circle are on the grey area. These special nodes change upstream traffic load over time while receiving 2 pkts/min from the server continuously. For the first 8 minutes, each node generates 2 pkts/min to the server as the other nodes. For the next 8 minutes, each node increases its upstream traffic load and generates 4 pkts/min to the



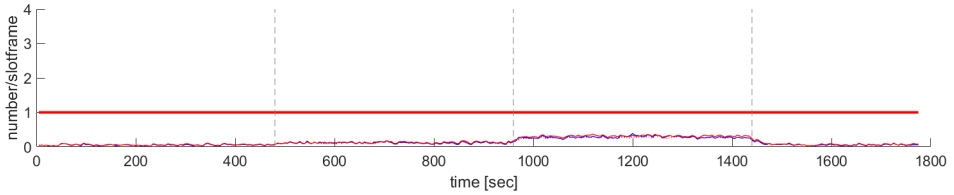
(a) RPL topology



(b) The number of cells allocation over time at the link (A, R)



(c) The number of cells allocation over time at the link (B, A)



(d) The number of cells allocation over time at the link (C, B)

Figure 8.2 Cell allocation by traffic load over time

Cell allocation of TRF-ALICE by time when the traffic load changes over time. The used unicast slotframe length is 40 and N_z=4. 21 nodes marked as red circle in the figure (a) generate different upward traffic load over time.

server. For the next 8 minutes, each node increases the upstream traffic load and generates 10 pkts/min to the server. For the last 6 minutes, it decreases the upstream traffic load and generates 2 pkts/min to the server as the other nodes.

During the experiment, upstream and downstream PDR were 99.89% and 99.92%, respectively. latency of upstream and downstream were 1,625ms and 1,351ms, respectively. The average radio duty cycle was 2.66%.

In the used RPL topology (figure 8.2(a)), 21 nodes are connected to node A and the node A forwards the packets to the root node R. We see the number of tx/rx cells scheduled for the link (A, R), link (B, A) and link (C, B) over time. The number of tx/rx cells allocated for each link are described in the figure 8.2(b), 8.2(c) and 8.2(d), respectively. In case of the link (A, R), the more traffic load is concentrated compared to the link (B, A). As a result, it requires more number of cells per slotframe. In the figures 8.2(b), 8.2(c) and 8.2(d), the average (EWMA) numbers of successful transmission and reception per slotframe are described as blue and red dotted lines, respectively, and they have almost same values. The average (EWMA) numbers of transmitter's transmission attempt and the receiver's estimated transmission attempt per slotframe are described as blue and red lines, respectively, and they have almost similar values. The receiver's estimation of the number of transmitter's transmission attempt shows quite high accuracy.

The number of tx/rx cells allocated for a slotframe over time are described with thick blue and red lines, respectively. By considering the target cell utilization τ and the number of cells change thresholds, the number of cells to allocate is decided. In the figure 8.2(b), the transmitter is node A and the receiver is the root node R. In the figure 8.2(c), the transmitter is node B and the receiver is node A. In the figure 8.2(d), the transmitter is node C and the receiver is node

B.

At around 480s, 21 nodes increases its traffic load from 2 pkts/min to 4 pkts/min. At this time, the links (C, B) and (B, A) do not need to increase the number of cells for that link (figure 8.2(c) and 8.2(d)) since the nodes B and C have only 10 and 3 nodes in its sub-tree, respectively. However, in case of the node A, it has 21 nodes in its sub-tree. As the traffic load increases about two times, it allocates additional cell for the link (A, R) increasing the number of transmission cells (figure 8.2(b)).

At around 960s, the traffic load increased from 4 pkts/min to 10 pkts/min and a link (A, R) increased the number of tx/rx cells per slotframe by doubling the number of cells from 2 to 4. In this time, the link (B, A) also increased the number of cells per slotframe from 1 to 2 to support the increased traffic load. The link (C, B) does not need to increase the number of cells per a slotframe since it has not much traffic load due to its small sub-network size.

At around 1,440s, 21 nodes decrease the upstream traffic load from 10 pkts/min to 2 pkts/min. Thereby, all the links reduce the number of tx/rx cells allocation until 1 based on the decreased traffic load.

TRF-ALICE supports dynamic traffic load with high reliability in efficent way by introducing *autonomous* and *"traffic load" -based* dynamic cell allocation method.

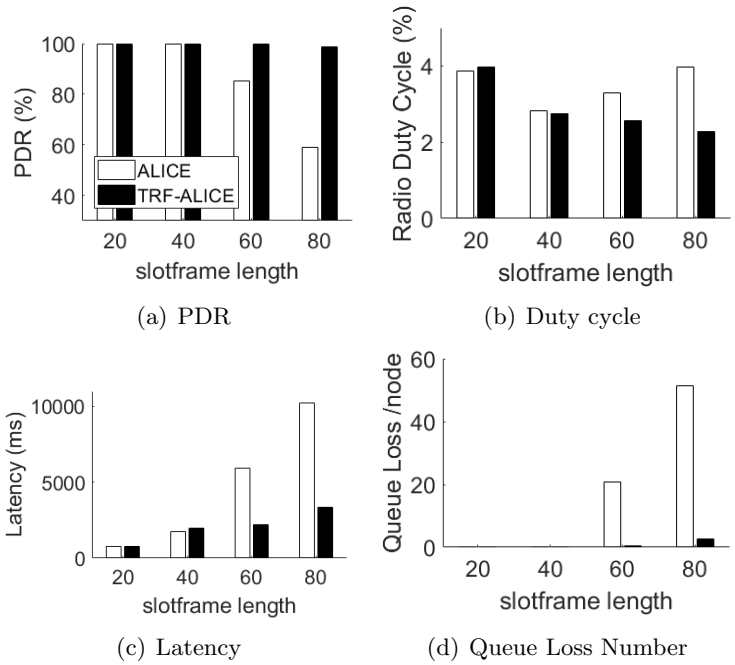
8.4 Impact of Unicast Slotframe Size on a Fixed Topology

We compare the performance of TRF-ALICE (N_z=4) and ALICE by the unicast slotframe size under the fixed topology as described in figure 8.2(a). When the

node switches its preferred parent on the RPL network, the connections between the nodes are not stable decreasing the communication performance. By fixing the RPL topology, we can see the effect of additive cell allocation more correctly.

Figure 8.3 describes the performance of ALICE and TRF-ALICE by the slotframe length. We increase the slotframe length from 20 to 80 and each node sends and receives 4 pkts/min to and from the server generating bidirectional traffic. In case of ALICE, it supports 99.9% PDR only when the slotframe length is 20 and 40. However, as the slotframe length increases, its PDR decreases significantly not supporting 4 pkts/min bidirectional traffic load (figure 8.3(a)). As the slotframe size increases, its queue level increases resulting long end-to-end latency ((figure 8.3(c)) and losing a lot of packets due to full transmission queue ((figure 8.3(d)). Important packets are lost in the network losing TSCH connection. Thereby, its radio duty cycle is increasing even with long slotframe length ((figure 8.3(b)). To be connected to the TSCH network, each node continuously listens the channel to receive TSCH EBs increasing its radio duty cycle significantly.

In case of TRF-ALICE, it supports high PDR regardless of the used slotframe size. Even with slotframe length 80, it supports 98.7% PDR (figure 8.3(a)). As the slotframe length increases, its radio duty cycle decreases supporting high PDR increasing energy efficiency (figure 8.3(b)). Even with long slotframe length, TRF-ALICE maintains short latency (figure 8.3(c)) and low queue loss number (figure 8.3(d)). Even when there are low communication opportunity due to long slotframe length, TRF-ALICE can increase the communication opportunity when it is needed based on the real-time traffic load for each link. Thereby, it could additionally allocate cells for the links with heavy traffic load. Especially for the links close to the root node. With this efficient and autonomous scheduling method, TRF-ALICE could achieve high



(a) PDR (b) Duty cycle

(c) Latency (d) Queue Loss Number

Figure 8.3 Performance by the slotframe length

Performance of ALICE and TRF-ALICE (N_z=4) according to unicast slotframe size under bidirectional traffic (4 pkts/min from each node and 4 pkts/min to each node). TRF-ALICE provides high PDR and energy efficiency even with large slotframe size.

communication performance even with a long slotframe length.

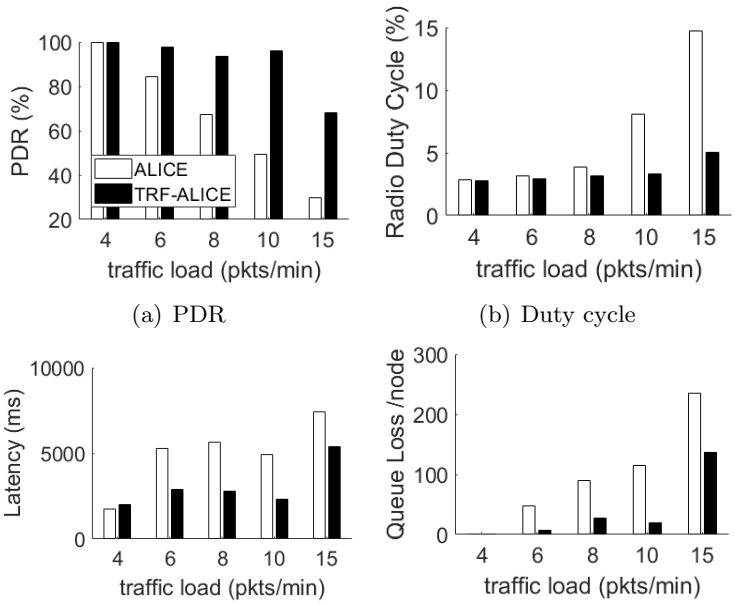
8.5 Impact of Traffic Load on a Fixed Topology

We compare the performance of TRF-ALICE (N_z=4) and ALICE by the traffic load under the fixed topology as described in figure 8.2(a). In this experiment, we use the slotframe size 40 for both schemes.

Figure 8.4 describes the performance of ALICE and TRF-ALICE by the traffic load when the slotframe size is 40. We increase the traffic load from 4 pkts/min to 15 pkts/min. At the x pkts/min traffic load, each node sends x pkts/min to the server and receives x pkts/min from the server generating 2x pkts/min on the network.

In case of ALICE, its PDR drastically decreases as the traffic load increases (figure 8.4(a)). Due to long transmission queue, each packet experiences long queueing delay resulting long end to end latency (figure 8.4(c)) and losing a large amount of packets due to full transmission queue (figure 8.4(d)). As it loses important packets, it also loses connection from the TSCH network and its radio duty cycle increases (figure 8.4(b)) continuously scanning the channel to receive TSCH EBs.

With TRF-ALICE, it supports high PDR (figure 8.4(a)), low latency (figure 8.4(c)) and low queue loss number (figure 8.4(d)) in energy efficient way (figure 8.4(b)) even with a heavy traffic load. When the traffic load increases, a node who has many packets to forward increases the number of cells for the links with a heavy traffic load. When a node needs a larger number of cells for a link, it can increase the number of tx/rx cells for that link by up to $N_z = 4$. Since additional cells are used only when they are needed in autonomous way, the radio duty cycle does not increase much. When the traffic load is 10 pkts/min,



(a) PDR

(b) Duty cycle

(c) Latency

(d) Queue Loss Number

Figure 8.4 Performance by the traffic load

Performance of ALICE and TRF-ALICE (N_z=4) according to traffic load when ALICE and TRF-ALICE use slotframe size 40.

TRF-ALICE supports 96.2% PDR, 2.3s end to end latency and 3.31% radio duty cycle. By allocating additional cells autonomously in efficient way, TRF-ALICE could improve the communication performance much more than that of ALICE in terms of PDR, latency and radio duty cycle.

Chapter 9

Conclusion

In this work, 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 var-

ious deployment scenarios. We also extended ALICE by considering the *real-time traffic load* without exchanging any information maintaining *autonomous* feature. The proposed method TRF-ALICE could achieve further performance improvements in terms of packet delivery ratio, latency and radio duty cycle compared to ALICE.

Bibliography

[1] 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.

[2] D.B. Johnson, Routing in Ad Hoc Networks of Mobile Hosts, 1994 First Workshop on Mobile Computing Systems and Applications, December 1994.

[3] Josh Broch, et al., A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, ACM/IEEE MobiCom 1998, 1998.

[4] S. Chen and K. Nahrstedt, Distributed QoS Routing in Ad Hoc Networks, IEEE Joumal on Selected Areas in Communications, vol. 17, no. 8, pp. 1488-1505, August 1999.

[5] Atis Elsts, et al., Enabling Healthcare in Smart Homes: The SPHERE IoT Network Infrastructure, IEEE Communications Magazine, December 2018.

[6] JeongGil Ko, Jong Hyun Lim, Yin Chen, Rvăzvan Musvaloiu-E, Andreas Terzis, Gerald M Masson, Tia Gao, Walt Destler, Leo Selavo, and Richard

P Dutton. 2010. MEDiSN: Medical emergency detection in sensor networks. ACM Trans. on Embedded Computing Systems (TECS) 10, 1 (2010), 11.

[7] Hyung-Sin Kim, JeongGil Ko, and Saewoong Bahk. 2017. Smarter markets for smarter life: applications, challenges, and deployment experiences. IEEE Communications Magazine 55, 5 (2017), 34–41.

[8] IEEE (Institute of Electrical and Electronics Engineers), https://www.ieee.org/

[9] IETF (Internet Engineering Task Force), https://www.ietf.org/

[10] 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).

[11] IETF 6TiSCH WG. https://datatracker.ietf.org/group/6tisch/about/.

[12] Thomas Watteyne, et al., 2010, Mitigating Multipath Fading Through Channel Hopping in Wireless Sensor Networks, IEEE ICC 2010.

[13] Chuan-Chin Pu and Wan-Young Chung, Mitigation of Multipath Fading Effects to Improve Indoor RSSI Performance, IEEE Sensors Journal, Vol.8, No.11, pp.1884-1886, 2008.

[14] Jamie S.C. Turner, et al., The study of human movement effect on Signal Strength for indoor WSN deployment, 2013 IEEE Conference on Wireless Sensor (ICWISE), December 2013.

[15] Jari Luomala and Ismo Hakala, Effects of Temperature and Humidity on Radio Signal Strength in Outdoor Wireless Sensor Networks, 2015 Feder-

ated Conference on Computer Science and Information Systems (FedCSIS), September 2015.

[16] Joakim Eriksson, et al., Scaling RPL to Dense and Large Networks with Constrained Memory, EWSN 2018.

[17] C. Bormann, M. Ersue and A. Keranen. May 2014. Terminology for Constrained-Node Networks. IETF, RFC 7228 (May 2014).

[18] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. 2002. Wireless sensor networks for habitat monitoring. In International workshop on Wireless sensor networks and applications. ACM, 88–97.

[19] Pangun Park, Carlo Fischione, Alvise Bonivento, Karl H Johansson, and Alberto Sangiovanni-Vincent. 2011. Breath: An adaptive protocol for industrial control applications using wireless sensor networks. IEEE Trans. on Mobile Computing 10, 6 (2011), 821–838.

[20] Nicola Accettura, Maria Rita Palattella, Gennaro Boggia, Luigi Alfredo Grieco, and Mischa Dohler. 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 (2013), 337–350.

[21] 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.

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

[23] Simone Brienza, Domenico De Guglielmo, Giuseppe Anastasi, Marco Conti, and Vincenzo Neri. 2013. Strategies for opti- mal MAC parameter setting in IEEE 802.15.4 wireless sensor networks: A performance comparison. IEEE ISCC (2013).

[24] 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).

[25] R. Daidone, G. Dini, and G. Anastasi. July 2014. On evaluating the performance impact of the IEEE 802.15.4 security sub-layer. Computer Communications 47 (July 2014), 65–76.

[26] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. 2004. Contiki-a lightweight and flexible operating system for tiny networked sensors. In Local Computer Networks, 2004. 29th Annual IEEE International Conference on. IEEE, 455–462.

[27] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust mesh networks through au- tonomously scheduled TSCH. ACM Conference on Embedded Networked Sensor Systems (2015).

[28] Atis E., Xenofon F., Simon D., George O., R. J P., and Ian C. 2017. Temperature-Resilient Time Synchronization for the Internet of Things. IEEE Trans. on Industrial Informatics (2017).

[29] Atis Elsts, Simon Duquennoy, Xenofon Fafoutis, George Oikonomou, Robert Piechocki, and Ian Craddock. 2016. Microsecond-Accuracy Time Synchronization Using the IEEE 802.15. 4 TSCH Protocol. In Local Com-

puter Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on. IEEE, 156–164.

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

[31] D. Guglielmo, S. Brienza, and G. Anastasi. August 15, 2016. IEEE 802.15.4e: a Survey. In ELSEVIER Computer Communications, Vol. 88. ELSEVIER, 1–24.

[32] D. De Guglielmo, F. Restuccia, G. Anastasi, M. Conti, and S. Das. 2016. Accurate and efficient modeling of 802.15.4 unslotted CSMA-CA through event chains computation. IEEE Trans. on Mobile Computing (2016).

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

[34] 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).

[35] Deokwoo Jung, Zhenjie Zhang, and Marianne Winslett. 2017. Vibration Analysis for IoT Enabled Predictive Maintenance. In Data Engineering (ICDE), 2017 IEEE 33rd International Conference on. IEEE, 1271–1282.

[36] Hyung-Sin Kim, Hongchan Kim, Jeongyeup Paek, and Saewoong Bahk. 2017. Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks. IEEE Trans. on Mobile Computing 16, 4 (2017), 964–979.

[37] S. Kim, S. Pakzad, D. Culler, James D., Gregory F., Steven G., and Martin T. 2007. Health Monitoring of Civil Infrastructures using Wireless Sensor Networks. In Information processing in sensor networks (IPSN). ACM, 254–263.

[38] P. Di Marco, C. Fischione, F. Santucci, and K.H. Johansson. 2014. Modeling IEEE 802.15.4 networks over fading channels. IEEE Trans. on Wireless Communication 13, 10 (2014), 5366–5381.

[39] M. Mohammad, X. Guo, and M.C. Chan. 2016. Oppcast: Exploiting Spatial and Channel Diversity for Robust Data Collection in Urban Environments. IPSN (2016).

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

[41] R. Musaloiu-E and A. Tezis. 2008. Minimizing the effect of wifi interference in 802.15.4 wireless sensor networks. International Journal of Sensor Networks 3, 1 (2008), 43–54.

[42] 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.

[43] Contiki OS. Open. URL. http://www.contiki-os.org. (Open).

[44] Maria Rita Palattella, Nicola Accettura, Mischa Dohler, Luigi Alfredo Grieco, and Gennaro Boggia. 2012. Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. IEEE 23rd International Symposium on PIMRC (2012).

[45] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel. 2016. On-the-fly bandwidth reservation for 6tisch wireless industrial networks. IEEE Sensors 16, 2 (2016), 555–560.

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

[47] Joseph Polastre, Jason Hill, and David Culler. 2004. Versatile low power media access for wireless sensor networks. In Proc. of the 2nd international conference on Embedded networked sensor systems. ACM, 95–107.

[48] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. Grieco. 2018. Autonomous and traffic-aware scheduling for TSCH networks. In Computer Networks, Vol. 135. ELSEVIER, 201–211.

[49] Soua Ridha, Pascale Minet, and Erwan Livolant. 2012. MODESA: an optimized multichannel slot assignment for raw data convergecast in wireless sensor networks. IEEE IPCCC (2012).

[50] C. Kishore Singh, A. Kumar, and P.M. Ameer. 2008. Performance evaluation of an IEEE 802.15.4 sensor network with a Star Topology. ACM Wireless Networks 14, 4 (2008), 543–568.

[51] 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 (LR-WPANs). IEEE (2006).

[52] Ridha Soua, Pascale Minet, and Erwan Livolant. October 2015. Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15.4e TSCH

networks. Trans. on Emerging Telecommunications Technologies 27, 4 (October 2015), 557–575.

[53] David Stanislowski, Xavier Vilajosana, Qin Wang, Thomas Watteyne, and Kristofer SJ Pister. 2014. Adaptive synchronization in IEEE802.15.4e networks. IEEE Trans. on Industrial Informatics 10, 1 (2014), 795–802.

[54] 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.

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

[56] P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. March 2012. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. IETF, RFC 6550 (March 2012).

[57] 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.

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

[59] Thomas Watteyne, Ankur Mehta, and Kris Pister. 2009. Reliability through frequency diversity: why channel hopping makes sense. In Proc. of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks. ACM, 116–123.

[60] T. Watteyne, M. Palattella, and L. Grieco. May 2015. Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement. IETF, RFC 7554 (May 2015).

[61] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven Glaser, and Kris Pister. 2012. OpenWSN: a standards-based low-power wireless development environment. Trans. on Emerging Telecommunications Technologies 23, 5 (2012), 480–493.

[62] K. Yedavalli and B. Krishnamachari. 2008. Enhancement of the IEEE 802.15.4 MAC protocol for scalable data collection in dense sensor networks. Proceedings of WiOPT (2008).

[63] 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. MDPI Sensors (June 2013).

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

[65] Taewon Suh et al., Electronic Shelf Lables: Prototype Development and Validation Using A Design Science Approach, Journal of Information Technology Management, Vol.29, No.4, 2018.

[66] IEEE Computer Society, 2003, IEEE Standards 802.15.4, 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 (LR-WPANs), 1 October 2003.

[67] X. Vilajosana, K. Pister, T. Watteyne. May 2017. Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration. IETF, RFC 8180 (May 2017).

[68] Atis Elsts, et al., Instant: A TSCH Schedule for Data Collection from Mobile Nodes, EWSN 2019.

[69] Rasool Tavakoli, et al., Topology Management and TSCH Scheduling for Low-Latency Convergecast in In-Vehicle WSNs, 2018, IEEE Transactions on Industrial Informatics, Vol.15, No.2, pp.1082-1093, February, 2019.

[70] A. Paventhan, et al., 2015, Experimental evaluation of IETF 6TiSCH in the context of Smart Grid, IEEE WF-IoT 2015.

[71] IETF Roll WG. https://datatracker.ietf.org/wg/roll/about/.

[72] Q. Wang, Ed., X. Vilajosana and T. Watteyne, November 2018. 6TiSCH Operation Sublayer (6top) Protocol (6P). IETF, RFC 8480 (November 2018).

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

[74] Hyung-Sin Kim et al., System architecture directions for post-soc/32-bit networked sensors, ACM SenSys 2018.

[75] Hyung-Sin Kim et al., Do not lose bandwidth: Adaptive transmission power and multihop topology control, IEEE DCOSS 2017, 2017.

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

[77] FIT/IoT-LAB public testbed, http://www.iot-lab.info.

[78] H.-S. Kim et al., Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey, IEEE Communications Surveys and Tutorials, Vol.19, No.4, pp.2502-2525, 2017.

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

[80] D. Dujouvne, et al., 6TiSCH 6top Scheduling Function Zero (SF0), IETF 6TiSCH Internet Draft, 5/12/2016-1/3/2018.

[81] T. Chang, et al., 6TiSCH Minimal Scheduling Function (MSF), IETF 6TiSCH Internet Draft, 8/21/2018-current (active I-D).

[82] JP. Vasseur, et al., March 2012, Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks, IETF Roll RFC 6551, March 2012.

[83] ALICE (Autonomous Link-based Cell Scheduling for TSCH) source code, https://github.com/skimskimskim/ALICE. (Open).

요약

TSCH(시간 슬롯 채널 호핑)는 IEEE 802.15.4e 표준이자 IETF 6TiSCH 연구 그룹의 핵심 프로토콜로서, TDMA 및 채널호핑 기술을 동시에 적용하여 간섭이 있는 환경에서도 높은 통신 신뢰성을 달성할 수 있게 하는 링크 계층 통신 기술이다. 신뢰적 TSCH 서비스 구현을 위해서는 잘 고안된 시간-채널 접근 스케쥴링 기술이 뒷받침 되어야 하는데, 기존 방식은 스케쥴을 위한 정보 교환에 많은 오버헤드를 야기하여 유동성, 범용성, 확장성 면에 제약이 있다. 최근 노드 정보 해쉬 기반의 자율적 스케쥴링 기법이 제안되었으나 트래픽이 많아지면 저조한 성능을 보인다는 한계가 있다. 본 연구는 자율적 링크 기반 스케쥴링 기술 ALICE를 소개한다. ALICE는 노드 간 통신 스케쥴을 시간 및 채널에 대해 고르게 분산시킴으로써 노드 간 경쟁을 완화하고, 더 많은 통신 기회를 효과적으로 할당하여 더 적은 에너지로 더 높은 통신 성능을 달성한다. 우리는 해당 기술을 국제 대규모 공공 테스트베드의 68개 노드에 구현하여 다양한 환경에서 다각도 성능 검증을 마쳤고, 최신 기법 대비 월등한 성능을 보임을 실험적으로 입증하였다. 특히 ALICE 기술의 강점은 트래픽이 많고 밀집된 네트워크에서 극대화되어 최신 기술 대비 두 배 이상의 패킷 처리량을 달성하면서도 98.3%의 높은 전송률을 확보하였고, 전송 지연 시간을 70%, 라디오 듀티 사이클을 35% 이상 감소시켰다. 뿐만 아니라 ALICE는 탄탄한 링크 계층 서비스를 제공함으로써 라우팅 계층에서의 RPL 부모 변경 횟수를 95% 이상 감소시키며, 상위 계층 프로토콜 동작 안정성까지 확보하였다. 우리는 ALICE 기법에 실시간 트래픽 정보를 반영하여 TRF-ALICE 기법으로 확장시켰다. 확장된 제안 기법 역시 노드 간 협상을 요구하지 않는 자율적 스케쥴링 기법으로, 각 노드는 트래픽 양이 많은 방향성 링크에 추가 셀을 할당할 수 있다. 이에 따라 TRF-ALICE는 AILCE 대비 트래픽이 극심한 상황에서 라디오 듀티 사이클을 증가시키기 않으면서도 두 배 이상의 전송률과 50% 미만의 전송 지연

성능을 달성하였다.

주요어: 802.15.4e, 시간 슬롯 채널 호핑, 스케줄링, 센서 네트워크, 사물인터넷
학번: 2013-20761