ECTS: Enhanced Centralized TSCH Scheduling with Packet Aggregation for Industrial IoT

Costas Michaelides [1], Toni Adame [1], and Boris Bellalta [1]

[1]Affiliation not available

October 30, 2023

Abstract

The Industrial Internet of Things (IoT) has gained a lot of momentum thanks to the introduction of Time Slotted Channel Hopping (TSCH) in IEEE 802.15.4. At last, we can enjoy collision-free, low-latency wireless communication in challenging environments. Nevertheless, the fixed size of time slots in TSCH provides an opportunity for further enhancements. In this paper, we propose an enhanced centralized TSCH scheduling (ECTS) algorithm with simple packet aggregation while collecting data over a tree topology. Having in mind that the payload of a sensor node is rather short, we attempt to put more than one payload in one packet. Thus, we occupy just one cell to forward them. We investigated the schedule compactness of ECTS in Matlab, and we evaluated its operation, after implementing it in Contiki-NG, using Cooja. Our results show that ECTS with packet aggregation outperforms TASA in terms of slotframe duration and imposes fairness among the nodes in terms of latency. A validation exercise using real motes confirms its successful operation in real deployments.

ECTS: Enhanced Centralized TSCH Scheduling with Packet Aggregation for Industrial IoT

Costas Michaelides, Toni Adame and Boris Bellalta
Department of Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Spain
Email: rstname.lastname@upf.edu

Abstract—The Industrial Internet of Things (IoT) has gained a lot of momentum thanks to the introduction of Time Slotted Channel Hopping (TSCH) in IEEE 802.15.4. At last, we can enjoy collision-free, low-latency wireless communication in challenging environments. Nevertheless, the xed size of time slots in TSCH provides an opportunity for further enhancements. In this paper, we propose an enhanced centralized TSCH scheduling (ECTS) algorithm with simple packet aggregation while collecting data over a tree topology. Having in mind that the payload of a sensor node is rather short, we attempt to put more than one payload in one packet. Thus, we occupy just one cell to forward them. We investigated the schedule compactness of ECTS in Matlab, and we evaluated its operation, after implementing it in Contiki-NG, using Cooja. Our results show that ECTS with packet aggregation outperforms TASA in terms of slotframe duration and imposes fairness among the nodes in terms of latency. A validation exercise using real motes con rms its successful operation in real deployments.

Index Terms—Convergecast, Industrial IoT, TSCH.

I. Introduction

In real-time Industrial IoT applications, such as the digital representation of complex systems, deterministic latency is of critical importance. As pointed out in [1], the main issue of complex systems is not that they are complex, but the fact that they may fail. In such cases, the timely collection of sensor data is our primary concern. The data packets are generally short, since they carry sensor values. However, industrial applications use a large number of sensor nodes and have stringent requirements concerning reliability and latency.

Time slotted channel hopping (TSCH), introduced in [2], is a MAC mode of IEEE 802.15.4 [3] targeted at industrial IoT applications. It allows reliable, interference-free links and offers deterministic latency, since every transmission can be scheduled. Its slotframe can be represented as a matrix. Each matrix cell corresponds to a time offset and a channel offset. We can foresee that there are many possibilities for scheduling, and a remarkable increase in capacity, in case it is properly implemented. Nonetheless, the use of TSCH in industrial environments requires a complete stack, up to the Application layer. The missing layers on top of IEEE 802.15.4 TSCH have been provided by the Internet Engineering Task Force (IETF) 6TiSCH [4] working group. It is a valuable work in progress which aims to bring IPv6 to industrial applications.

In this work, we propose an enhanced centralized scheduling TSCH (ECTS) algorithm targeted at Industrial IoT, that can be easily implemented. We aim to reduce data packet transmissions [5] and enhance the cooperation among the sensor nodes [6]. Our concept is simple. Since the size of a time slot is xed, we may allow a node to aggregate a number of packets, add their payloads in one packet and forward it to the next node occupying just one cell. Fig. 1 depicts an example. On the left there is a node arrangement in tree topology and on the right there is the TSCH matrix. In the rst row, nodes E and F transmit their packets sequentially to node C. Then, node C aggregates both payloads along with its own and forwards one packet to node A. This is a very simple way to take advantage of unused airtime in each cell and reduce latency thanks to a neat compact schedule.

Fig. 1. An example of ECTS: In the highlighted cell, node C sends a data packet to node A that includes its own payload and the payloads of E and F.

To the best of our knowledge, this is the rst work which focuses on packet aggregation in TSCH and attempts to draw some conclusions about the potential bene ts. We investigated the schedule compactness of ECTS in Matlab and then, after implementing ECTS using Contiki-NG [7], we evaluated its performance in Cooja, the emulator. Our ndings are summarized in the following:

1) The state-of-the-art can be outperformed in terms of schedule compactness using packet aggregation,
2) Packet aggregation imposes fairness among the nodes in terms of latency.

Also, ECTS works as expected on our Zolertia [8] testbed, leaving an extensive on- eld validation for future work.

The rest of this paper is organized as follows: Section II provides an overview of TSCH in Industrial IoT. The proposed ECTS algorithm is presented in Section III. Its performance evaluation in Matlab and Contiki-NG is included in Section IV. In Section V there is a short note on our rst tests with real devices. Finally, Section VI concludes this paper.

II. TSCH Scheduling in Industrial IoT

TSCH is a powerful MAC mode of IEEE 802.15.4 that is primarily useful in challenging environments. In this chapter we review the basics of TSCH and provide an overview of the scheduling approaches in Industrial IoT.

A. Preliminaries

Industrial applications relied on well-established wired technologies for a long time. WirelessHART [9], a breakthrough at its time, introduced reliable IEEE 802.15.4 based communications and served as a guideline for the specification of TSCH in IEEE 802.15.4. Its basic principles can be easily identified in the specification of TSCH.

TSCH provides diversity in time and frequency, which is usually depicted in 2D as a slotframe, using time offsets on X axis and channel offsets on Y axis. The slotframe is repeated over time. A pair of a time offset and a channel offset is defined as a cell. We also use the term *time slot*, with reference to the X axis. The typical duration of a time slot is from 10 ms to 15 ms. It is enough for the transmission of a data packet and its acknowledgement. The standard defines a TSCH coordinator and enhanced beacons (EB) for the delivery of the essential information to the nodes. Upon initialization, the TSCH coordinator sets the absolute slot number (ASN) to zero, and the frequency hopping sequence (FHS). Therefore, the physical channel (CH) of a time slot is calculated by:

$$CH = FHS[(ASN + channelOffset) \bmod \|FHS\|]. \quad (1)$$

Today, scheduling in multiple frequencies is universally praised. We would like to highlight [10], an early work on this topic that identified the opportunities of using multiple frequencies. The aforementioned work studies convergecast, i.e. the collection of data over a tree topology, and provides the theoretical bounds of aggregated and raw-data convergecast.

B. Scheduling algorithms

TSCH scheduling is a thriving field of research. The scheduling algorithms can be classified as centralized or decentralized. Moreover, decentralized algorithms can be either distributed or autonomous. Centralized algorithms are targeted at static applications, while decentralized ones are more suitable for dynamic applications. Centralization has been identified in [11] as beneficial in terms of delay. However, the decentralized approaches are usually less complex and scale up effortlessly.

A centralized algorithm, such as the pioneering TASA [12], has perfect knowledge of the topology and builds the schedule at the coordinator. A distributed algorithm, such as DeTAS [13], allows the negotiation among neighbor nodes. The autonomous approach, which is particularly useful in mesh networks, does not require any negotiations among the nodes. Each node builds its own schedule in inventive ways. See Orchestra [14], a valuable work on this topic. On top of scheduling we may also use blacklisting or whitelisting [15], in order to select the best available channel offsets and improve reliability.

Packet aggregation in TSCH has been already identified as beneficial in the literature. AMUS [16] uses centralized scheduling to aggregate packets and relay them, in order to reduce the delay. LaDiS [17] is a distributed scheduler, where the schedule of each node is constructed by its parent. Both AMUS and LaDiS take into account that children should be scheduled before their parents in order to benefit from packet aggregation. The authors of LLTT [18] acknowledge this as well, but propose a practical periodic aggregation at the network layer instead. They also use shared time slots in order to boost the performance of LLTT. OST [19] combines several scheduling approaches. Its authors investigate several traffic patterns, and consider packet aggregation both in the uplink and downlink.

In our work, we aim to draw some clear conclusions regarding the operation of packet aggregation and propose how to put it in practice. Like previous works, we confirm that the children have to be scheduled to transmit before their parents. Additionally, we propose an enhanced centralized scheduling algorithm, and we show that moderate use of packet aggregation allows us to outperform the state-of-the-art. We will elaborate on this in the following chapters.

III. Enhanced Centralized TSCH Scheduling

In this section we introduce the proposed ECTS algorithm and present a comprehensive example based on Fig. 1, in order to showcase its operation.

A. Motivation

Our focus is on Industrial IoT in tree topology, where there is a large number of sensor nodes with short payloads. We aim to take advantage of the tree structure, aggregate packets in order to reduce the transmissions and build a compact schedule. The proposed ECTS algorithm offers a simple way of collecting short packets over a tree topology. Each parent aggregates the data packets of its children and combines their payloads in one or more packets without any further process. We aim to investigate the operation of packet aggregation and identify when this is actually beneficial, particularly in terms of schedule compactness and latency.

B. Description

ECTS is a scheduling algorithm for convergecast. It is a centralized approach, since the network topology in Industrial IoT is expected to be more or less static. We follow three simple rules:

1) A node can be a child, a parent, or both,
2) A child forwards its packets to its parent,
3) A parent aggregates the packets of its children.

First, we build a tree structure and establish the relationships between the nodes. The nodes are sorted according to their distance from the coordinator, and a parent is selected for each child according to its RSSI. Next, we build a node list for each time slot that includes the eligible nodes for transmission. We start with the leaf nodes and move upwards in each step until we reach the coordinator. Also, we make sure that the children

Fig. 2. Flowchart of the ECTS algorithm.

have an opportunity to transmit before their parents, so the parents can aggregate the packets of their childern.

During the construction of the schedule, we have to deal with three major constraints:

1) Interference,
2) Half-duplex transceivers,
3) Topology.

Our cells accommodate only one link, in order to guarantee that the scheduled links are free of interference from other transmissions in the same network. However, we have to consider the other two constraints: the half-duplex transceivers and the topology. Inevitably, we assume that our nodes are half-duplex. Therefore, a node can establish only one connection during a time slot, either for transmission or reception. Regarding topology, we would love to have perfectly balanced sub-trees and assign an exclusive channel offset to each one, but this rarely happens. Topology is actually the most important constraint during data collection.

The ECTS algorithm is depicted in Fig. 2. The schedule is constructed by the coordinator and disseminated to the nodes using the first two time slots of each slotframe. First, we create a node list for each time slot and select a random node which has a packet to send. Next, we confirm that it is not scheduled as a sender or a receiver in any other cell of the current time slot. Then, we check if there is an available channel offset. If the answer is positive, a cell is scheduled for this node. In any case, the algorithm proceeds to the next time slot.

We continue with an illustrative example of ECTS, based on Fig. 1. In the following, S is the slotframe of size 3×3 and each one of its elements s_{ij} represents a scheduled link at channel offset i and time offset j. Also, T_j is the time slot at time offset j. Thus, a slotframe can be represented as

$$S = \begin{bmatrix} T_1 & T_2 & T_3 \end{bmatrix}. \tag{2}$$

Each node generates one packet per slotframe. Each cell can accommodate one link. Node A, the root of the tree, is the coordinator. The node list has been already constructed, and the children have been prioritized over their parents. We proceed to the next step of the flowchart in Fig. 2, in order to start scheduling the nodes in each time slot.

In the first time slot, there are three eligible nodes to be scheduled: D, E, F. All of them are leaf nodes, so we pick one randomly, for example E. E is a child of C, thus $s_{11} = E \rightarrow C$. Next, we pick F, but unfortunately it cannot be scheduled since its parent C has been already scheduled for reception in this time slot. Next, we pick D and schedule it at the second channel offset $s_{21} = D \rightarrow B$. Thus, the first time slot is

$$T_1 = \begin{bmatrix} E \rightarrow C \\ D \rightarrow B \\ \emptyset \end{bmatrix}. \tag{3}$$

In the second time slot there are more eligible nodes: B, D, E, F. Note that node C is not eligible since one of its children has not been scheduled yet. We pick randomly F, so we set $s_{12} = F \rightarrow C$. We pick E, but it does not have any packets to send. The same is true for D. Then, we pick B and set $s_{22} = B \rightarrow A$. Thus, the second time slot is

$$T_2 = \begin{bmatrix} F \rightarrow C \\ B \rightarrow A \\ \emptyset \end{bmatrix}. \tag{4}$$

In the third time slot there are even more eligible nodes: B, C, D, E, F. However, no node apart from C has any packets to send. Eventually, we can set only $s_{13} = C \rightarrow A$. Thus, the third time slot is

$$T_3 = \begin{bmatrix} C \rightarrow A \\ \emptyset \\ \emptyset \end{bmatrix}. \tag{5}$$

Finally, the constructed schedule of the slotframe is:

$$S = \begin{bmatrix} E \rightarrow C & F \rightarrow C & C \rightarrow A \\ D \rightarrow B & B \rightarrow A & \emptyset \\ \emptyset & \emptyset & \emptyset \end{bmatrix}. \tag{6}$$

In the aforementioned example, we were fortunate enough to deliver every generated packet to the coordinator. Moreover, we achieved a remarkably compact schedule, since $s_{13} = C \rightarrow A$ delivers the payloads of three nodes (C, F and E) to the coordinator at once. Thus, there is an improvement in terms of latency. Also, note that we have not reached the maximum channel offsets. Despite the fact that our example is simple, this observation is valid. Most of the times we are restricted either by the half-duplex transceivers or by the topology. We will return to this discussion in Chapter IV.

TABLE I
SIMULATION PARAMETERS IN MATLAB.

	Parameter	Value
Scenario	Realizations	1000
	Area	200 m 200 m
	Coordinator	1
	Nodes	from 5 to 50
	Node deployment	Uniform
	Packet rate per node	1 packet/slotframe
MAC	MSDU	102 bytes
	Payload	25 bytes
	Channel offsets	2, 4, 8, 16
	Maximum aggregated packets	4
PHY	Frequency	2.4 GHz
Channel	Pathloss model	Lognormal (ITU-R)

Fig. 3. Random node deployment of 50 nodes in Matlab.

IV. PERFORMANCE EVALUATION

Our performance evaluation consists of simulations in Matlab and Contiki-NG. First, we investigate the schedule compactness of ECTS in Matlab. Next, we apply our findings in Contiki-NG simulations using Cooja.

A. Simulation in Matlab

We used Matlab to test the performance of the proposed ECTS algorithm. Our baseline is TASA. The theoretical bounds of TASA were provided by its authors in [12]. The nodes are uniformly distributed in an area of 200 m 200 m and the coordinator is placed in the center. See Fig. 3 for an example with 50 nodes. Each node generates one packet in each slotframe. Please see Table I for more insight.

In compliance with IEEE 802.15.4, the maximum size of our frame is 127 bytes. We reserve 25 bytes for the MAC header and use 102 bytes for the MAC service data unit (MSDU). When we allow packet aggregation, our payloads should fit in MSDU. In our case, i.e. up to four aggregated packets, the payloads are 25 bytes each. We assume that all transmissions are successfully delivered, since our focus is on the evaluation of the constructed schedule in terms of compactness.

Fig. 4. Minimum slotframe duration using four channel offsets (Matlab).

Fig. 5. Minimum slotframe duration for 50 nodes (Matlab).

Fig. 4 shows the slotframe duration for TASA and ECTS when we use up to four channel offsets. We notice that ECTS without packet aggregation performs well just for a few nodes, but it scales worse than TASA. Indeed, the algorithm relies on packet aggregation. Thus, when we allow up to four aggregated payloads in each data packet, ECTS achieves a shorter slotframe than TASA.

Fig. 5 shows the slotframe duration for 50 nodes, using up to two, four, eight and 16 channel offsets. We confirm that there is an improvement when we allow up to four channel offsets. However, if we allow more than four channel offsets the improvement is incremental. This happens because the performance is bounded by the topology. That is, the structure of the tree and its sub-trees does not allow any significant improvement. TASA does not improve as well, because its performance depends on the traffic load [12].

Our results in Matlab show that by allowing up to four aggregated packets and up to four channel offsets we outperform TASA marginally. An increase of channel offsets provides further improvement, but it depends on our tree structure. This is an interesting topic for further research as well.

TABLE II
SIMULATION PARAMETERS IN COOJA.

	Parameter	Value
Scenario	Time	10 minutes
	Coordinator	1
	Nodes	15 (see Fig. 6)
	Packet rate per node	1 packet/s
NET	Routing	RPL Lite
MAC	MSDU	102 bytes
	Payload	25 bytes
	Slotframe size	20 time slots
	Time slot duration	10 ms
	Channel offsets	4
	Maximum aggregated packets	4
PHY	Frequency	2.4 GHz
Channel	Model	Unit Disk Graph

Fig. 6. The implemented scenario in Cooja.

B. Simulation in Contiki-NG with Cooja

We implemented ECTS in Contiki-NG and evaluated its operation using several node deployments in Cooja. We would like to share some interesting results using the CSMA/CA mode of Contiki-NG as a baseline. The depicted network in Fig. 6 consists of one coordinator and 15 nodes in tree topology. The slotframe has 20 time slots and each time slot has a duration of 10 ms. The payload is 25 bytes. Please see Table II for the simulation parameters.

Based on our findings in Matlab, we allow up to four channel offsets and up to four aggregated packets. Our slotframe here has 20 time slots. This is not the optimal slotframe size for ECTS, but it is nonetheless a reasonable choice, since we can accommodate an adequate number of transmissions and have a fair comparison with CSMA/CA. Next, we present some indicative results, captured after the dissemination of the schedule, in order to highlight the benefits of ECTS in terms of latency.

Fig. 7. Packet delay for each node using CSMA/CA (Cooja).

Fig. 8. Packet delay for each node using ECTS (Cooja).

Fig. 7 shows the latency for each node when all nodes use CSMA/CA at the link-layer, and Fig. 8 shows the latency for each node using ECTS. While CSMA/CA has very good results, the results of ECTS are much more interesting. They show that every node enjoys a deterministic, low latency, about 50 ms delay. Thanks to packet aggregation, ECTS imposes fairness in terms of latency. This is an intuitive result, since we group more than one payload and transmit them together.

The exact amount of latency using ECTS depends on the slotframe size. For example, see the outliers in Fig. 8. We notice that some children of node 2 suffer from an occasional additional delay. This happens in the unfortunate case where a packet is not delivered for any reason, e.g. a transmission error. In such cases, the next opportunity for transmission is in the next slotframe. Thus, there is an additional delay of 20 slots × 10 ms = 200 ms. If a packet contains more than one payload, this delay affects more than one node.

We presented some indicative results to illustrate the strengths and weaknesses of ECTS. Our results in Cooja are for a specific configuration, hopefully enlightening. We noticed that the structure of the tree if of great importance. Actually, this may serve as a motivation for further research, in order to build more balanced trees in terms of traffic and topology. We believe that the enforcement of fairness suits Industrial IoT well. A real-time industrial system is evaluated as a whole, not by the performance of each sensor node separately!

Fig. 9. A snapshot while testing ECTS on RE-Motes.

V. Preliminary Tests

As this work is a part of an ongoing project, Looming Factory, we would like to report that the rst experimental trials on our Zolertia based testbed were successful. Fig. 9 shows one of our tests. We use an Orion Router as the coordinator and RE-Motes as nodes. A larger scale test is being prepared by our project partners, in order to collect sensor data in an industrial environment.

The ECTS implementation used in our trials is the same that has been validated with Cooja, so the results were more or less as expected. However, in our implementation, there is still much room for improvement regarding the dissemination of the schedule, which currently occupies two time slots of the slotframe.

VI. Conclusion

ECTS targets industrial applications in tree topology and exploits both TSCH and packet aggregation. We achieved some remarkable results in terms of schedule compactness and latency, by allowing up to four channel offsets and up to four aggregated packets. Our only assumption is that parent nodes are willing to help their children. Eventually, it appears that this is bene cial for every node. We believe that ECTS is a feasible, easily implemented scheduling approach for the future real-time Industrial IoT.

Acknowledgments

The authors would like to thank Fernando Maura Rivero for taking photos during the tests on RE-Motes. Also, the authors would like to thank the anonymous reviewers for their detailed and constructive comments. This work was supported in part by grant WINDMAL PGC2018-099959-B-I00 (MCIU/AEI/FEDER,UE), and co- nanced by the European Union Regional Development Fund within the framework of the ERDF Operational Program of Catalonia 2014–2020 (Looming Factory).

References

[1] M. Grieves and J. Vickers, Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. Cham: Springer International Publishing, 2017, pp. 85–113.

[2] IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer, IEEE Std. 802.15.4e, July 2012.

[3] IEEE Standard for Low-Rate Wireless Networks, IEEE Std. 802.15.4, July 2020.

[4] X. Vilajosana, T. Watteyne, T. Chang, M. Vučinić, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A Tutorial," IEEE Communications Surveys Tutorials, vol. 22, no. 1, pp. 595–615, 2020.

[5] G. M. Dias, B. Bellalta, and S. Oechsner, "A Survey About Prediction-Based Data Reduction in Wireless Sensor Networks," ACM Comput. Surv. vol. 49, no. 3, Nov. 2016. [Online]. Available: https://doi.org/10.1145/2996356

[6] C. Michaelides and F.-N. Pavlidou, "Mutual Aid Among Sensors: An Emergency Function for Sensor Networks," IEEE Sensors Letters, vol. 4, no. 9, pp. 1–4, 2020.

[7] Contiki-NG. [Online]. Available: https://www.contiki-ng.org/

[8] Zolertia. [Online]. Available: https://zolertia.io/

[9] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," in 2008 IEEE Real-Time and Embedded Technology and Applications Symposium, 2008, pp. 377–386.

[10] O. Durmaz Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast Data Collection in Tree-Based Wireless Sensor Networks," IEEE Transactions on Mobile Computing, vol. 11, no. 1, pp. 86–99, 2012.

[11] J. N. Tsitsiklis and K. Xu, "On the Power of (Even a Little) Centralization in Distributed Processing," SIGMETRICS Perform. Eval. Rev, vol. 39, no. 1, pp. 121–132, Jun. 2011. [Online]. Available: https://doi.org/10.1145/2007116.2007131

[12] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH," IEEE Sensors Journal, vol. 13, no. 10, pp. 3655–3666, 2013.

[13] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized Traf c Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation," IEEE Internet of Things Journal, vol. 2, no. 6, pp. 455–470, 2015.

[14] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, ser. SenSys '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 337–350. [Online]. Available: https://doi.org/10.1145/2809695.2809714

[15] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "Whitelisting Without Collisions for Centralized Scheduling in Wireless Industrial Networks," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5713–5721, 2019.

[16] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks," in 2016 IEEE Wireless Communications and Networking Conference, 2016, pp. 1–6.

[17] H. Hajian, M. Nabi, M. Fakouri, and F. Veisi, "LaDiS: a Low-Latency Distributed Scheduler for Time-Slotted Channel Hopping Networks," in 2019 IEEE Wireless Communications and Networking Conference (WCNC) 2019, pp. 1–7.

[18] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Topology Management and TSCH Scheduling for Low-Latency Convergecast in In-Vehicle WSNs," IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 1082–1093, 2019.

[19] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "OST: On-Demand TSCH Scheduling with Traf c-Awareness," in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 69–78.