Competition: Centrally Scheduled Low-Power Wireless Networking for Dependable Data Collection

Oliver Harms[†], Olaf Landsiedel[‡†]
[†] Chalmers University of Technology, Sweden
[‡] Kiel University, Germany
harms@chalmers.se, ol@informatik.uni-kiel.de

Abstract

For low-power wireless networks, it is important to survive interference to be usable for Industrial Internet-of-Things (IIoT) applications. Distributed flooding protocols like Glossy or Chaos have shown that they can meet the expectations of surviving interference and node failures. However, non-distributed, centralized schedulers are favorable for IIoT but are not used yet in these environments. In this paper, we explore the use of centralized schedulers for low-power wireless networks to achieve robustness in data collection applications.

1 Introduction

The growing number of Internet of Things (IoT) and particularly Industrial Internet of Things (IIoT) applications using wireless communication has strong requirements on reliability and latency. Because many devices are battery powered, or harvest their energy themselves, low-power operation including a low energy consumption is another important factor to keep as low as possible.

Based on the mentioned requirements, different protocols for data collection, especially distributed ones, have been developed. These include Glossy [3] and Glossy-based protocols as well as protocols based on the best-effort routing protocol RPL [7]. These protocols have the advantage to be able to adapt easily to non-predefined communications. A centralized approach on the other hand, needs to know which communications have to be performed at a given point in time and can be able to guarantee a certain reliability and latency for these communications. Moreover, centralized approaches are not necessarily dependant on routing protocol like RPL. Existing centralized schedulers focus mainly on the schedulability of communications in (almost) interference-free networks with highly reliable links.

We present C-TSCH, a centralized scheduler for TSCH, working with a novel routing strategy *sliding windows*. This strategy enables the use of centralized schedulers in networks susceptible to interference, combining the advantages of centralized schedulers with those of traditionally distributed ones.

International Conference on Embedded Wireless Systems and Networks (EWSN) 2019
25–27 February, Beijing, China

ISBN: 978-0-9949886-3-8

2 Centralized Scheduler (C-TSCH)

The centralized scheduler we present in this paper is based on the TSCH layer implemented in Contiki-NG [5, 2, 1]. Contrary to previously presented centralized scheduling algorithms, like C-LLF [6], focusing mainly on schedulability, the focus of our scheduler is on reliability of communications in networks susceptible to interference.

The scheduler consists of two modules, a Contiki network layer and a scheduling software, on a central server, performing the routing and scheduling using a modified version of a shortest-path first scheduler and applying the retransmission strategy of sliding windows to it. The network layer performs the neighbor discovery, the implementation of the TSCH schedule as well as the main communication functions of sending, receiving and forwarding. The neighbor discovery for the competition version of the scheduler was performed prior to the evaluation using broadcasts with individual transmission slots for each node in the network.

Besides these network layer implementations, the following changes of the TSCH layer were necessary. We extended the behaviour of the TSCH queues from neighbor-based to flow-based. Flow-based queues have the advantage of being able to forward data independent of the order it was received in.

The scheduling software which is written in Python allows an easier implementation and testing of different scheduling algorithms and retransmission strategies than a scheduler which is part of the nodes' operating system. Our scheduling software creates the schedules for the node layouts given for the competition, based on the data from multiple neighbor discovery runs. It uses a modified shortest-path first scheduler with the retransmission strategy of sliding windows and generates the schedule as C-code which is used for the evaluation of the competition.

2.1 Sliding Windows

Our scheduling strategy of sliding windows introduces a variable number of retransmissions for a route from a transmitter to a receiver to add robustness of the system to interference while adding the least possible increase of latency.

	1	2	3	4	5	6	7
A	TX	TX	TX	TX			
B	RX	RXTX	RXTX	RXTX	TX		
C		RX	RXTX	RXTX	RXTX	TX	
D			RX	RXTX	RXTX	RXTX	TX
E				RX	RX	RX	RX

Figure 1. An example of the sliding window strategy. A packet is to be transmitted from node A to E via B, C and D. The size of the sliding window is 5, as seen for nodes B, C and D.

To have a variable number of retransmissions per hop of a communication flow, we use TSCH slots that can be either used for reception or for transmission (RXTX). With these slots, our scheduler is not limited to a fixed number of retransmissions per hop but can instead dynamically perform as many retransmissions as needed depending on the surrounding interference level. After a successful transmission, the remaining transmission slots are skipped.

An important parameter of this strategy is the window size which is the number of timeslots a node is maximally involved in for communication. This window size also determines how many timeslots a transmission maximally takes. In the best case, without interference, it takes no more timeslots than the number of hops, like traditional scheduling algorithms. Whereas it can take a maximum of 2 less than the window size additional timeslots to perform the end-to-end transmission.

Figure 1 shows a possible schedule for a route of 5 nodes with a window size of 5. For nodes B, C and D, it can be seen that those three nodes are involved in the communication for a maximum of 5 timeslots and have a radio on time of maximally 5 slots as well. Every node which might be sending or receiving in the same timeslot (marked with RXTX) has a shared slot of both modes executing one or the other based on the previous reception of the packet sent in this communication.

3 Design

The scheduler design for the competition consists of a shortest-path first scheduler in combination with the retransmission strategy of sliding windows. We computed the schedules for the given scenarios and the given network topology beforehand. Nodes that are not scheduled to be part of any of the communication flows for each scenario are not used and therefore their processor is set to low-power mode 4 (LPM 4) to save energy.

4 Assumptions

To be able to compete with our presented scheduler in the competition, we made the following assumption. We will perform the scheduling of the connections beforehand resulting in a set of static paths, therefore the source and destination nodes have to be known beforehand and must not change between runs.

5 Initial Results

The experiments we performed during the preparation phase show, that our scheduler is able to reach a reliability of almost 100% with an average latency of 455.5 ms and using only 160 J of energy for the whole testbed consisting of 51 nodes. These measurements were reached without any generated interference. Figure 2 shows the achievable reliability for different interference levels dependant on the acceptable latency. It shows that the created schedule performs well for no interference, for which the schedule is optimized, but there is room for improvements for scenarios with interference present.

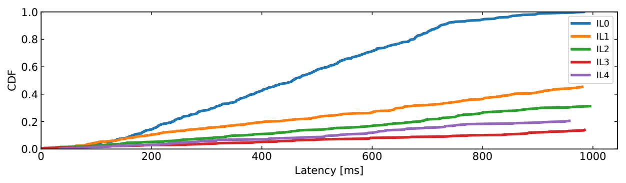

Figure 2. Reliability of our scheduler for all possible latencies for topology 1 with interference levels 0 to 4.

6 Discussion

As seen in the initial results, our presented solution works well for no or only little interference, but not well enough for IIoT applications susceptible to higher interference levels. This is due to using a single path only, for each communication flow, which cannot adapt to strong interference levels along the path. Moreover, the TSCH initialization takes a long time with high interference levels, leading to a low reliability in the beginning of an experimental run. Furthermore, the limited hardware capabilities of TelosB nodes (RAM and ROM) coupled with a modern operating system (Contiki-NG) and the need to include all possible schedules in the firmware are very challenging.

7 Conclusion & Future Work

In this paper, we present C-TSCH, a novel centralized scheduler based on sliding windows as a solution for the problem of data collection in low-power wireless networks susceptible to interference. For a future version of the scheduler, we plan to include a multi-path routing strategy as well as multicast transmissions to ensure better performance under interference.

8 Acknowledgments

This work was supported by the Swedish Foundation for Strategic Research (SSF) through the project LoWi, reference FFL15-0062.

9 References

[1] Contiki-NG: The OS for Next Generation IoT Devices. http://contiki-ng.org/.

[2] S. Duquennoy et al. TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation. In *IEEE DCOSS*, 2017.

[3] F. Ferrari et al. Efficient network flooding and time synchronization with Glossy. In *ACM/IEEE IPSN*, 2011.

[4] O. Landsiedel et al. Chaos: Versatile and Efficient All-to-All Data Sharing and In-Network Processing at Scale. In *ACM SenSys*, 2013.

[5] M. Palattella et al. Using IEEE 802.15. 4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement. 2015.

[6] A. Saifullah et al. Real-Time Scheduling for WirelessHART Networks. In *IEEE RTSS*, 2010.

[7] T. Winter et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Technical report, 2012.