Towards Delay-Minimal Scheduling through Reinforcement Learning in IEEE 802.15.4 DSME

1st Florian Meyer
Institute of Telematics, Hamburg University of Technology
Hamburg, Germany, fl.meyer@tuhh.de

2nd Volker Turau
Institute of Telematics, Hamburg University of Technology
Hamburg, Germany, turau@tuhh.de

***Abstract*—The rise of wireless sensor networks (WSNs) in industrial applications imposes novel demands on existing wireless protocols. The *deterministic and synchronous multi-channel extension* (DSME) is a recent amendment to the IEEE 802.15.4 standard, which aims for highly reliable, deterministic traffic in these industrial environments. It offers TDMA-based channel access, where slots are allocated in a distributed manner. In this work, we propose a novel scheduling algorithm for DSME which minimizes the delay in time-critical applications by employing reinforcement learning (RL) on deep neural networks (DNN).**

I. Introduction

Recently, WSNs have experienced an increased adoption in industrial applications due to their ease of deployment and cost-efficiency. These applications place strict requirements on the reliability and timeliness of the wireless transmission protocol. Therefore, the IEEE 802.15.4 standard was extended by a number of substandards, DSME amongst others, in 2015 to increase robustness and determinism in industrial environments [1]. DSME offers MF-TDMA-based channel access and features a distributed slot negotiation mechanism. Therefore, distributed scheduling is a natural choice for DSME. *Traffic-Aware and Predictive Scheduling* (TPS) is the default distributed scheduler for openDSME, an open-source implementation of DSME [2]. It aims for maximum reliability by utilizing an exponential moving average and a hysteresis for overprovisioning. However, it is unsuitable for time-critical applications because slots are placed randomly, i.e., packets might be queued some time before being transmitted. Additionally, TPS does not consider the level of the queue but only incoming flows. Queues can grow large, resulting in high end-to-end delays. Therefore, this work proposes a scheduling mechanism, utilizing machine learning, to minimize end-to-end delay while maintaining a reliability similar to TPS.

II. DSME in a Nutshell

In DSME, time is divided into a *contention-access period* (CAP) and a *contention-free period* (CFP), as depicted in Fig. 1. In the CAP, nodes communicate using CSMA/CA. The CFP is further subdivided into *guaranteed time slots* (GTS), which are spread over time and frequency and grant exclusive access to the shared medium. Initially, GTS must be negotiated between two nodes in a distributed manner using a 3-way handshake during the CAP. The combination of a CAP with 9 time slots and a CFP with 7 time slots is called *superframe* (SF). Multiple SFs are joined in a *multi-superframe* (MSF), after which a schedule of allocated GTS repeats [1].

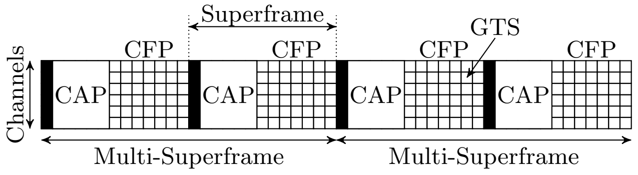

Fig. 1. Frame structure of IEEE 802.15.4 DSME.

III. Related Work

Liu et al. propose RL-MAC, a MAC-protocol where the active time and sleep time of nodes is dynamically adapted based on information inferred from other nodes in the network [3]. They achieve high throughput and low power consumption with mixed traffic conditions. Similarly, [4] proposes a MF-TDMA-based scheduler that avoids interference between different wireless protocols. They show that Neural Networks can accurately predict free slots in these scenarios.

IV. Formal Problem Description

In the following, we consider converge-cast scenarios in WSNs with arbitrary routing trees and N nodes $v_0, \dots, v_{N-1}$. Here, v_0 acts as the sink and every node $v_{i\neq 0}$ generates δ_i packets per second and transmits them towards v_0. For this, every second is divided into K time slots $k_0, \dots, k_{K-1}$ with equal length $t = \frac{1s}{K}$. The generation time τ_j of every packet p_j is know and lies within a time slot k_i. A packet generated in k_i can be transmitted in k_{i+1} at earliest.

A schedule θ is the assignment of a state from the set $\{$IDLE, TX, RX$\}$ to every slot k_{ij} of node v_j. During a TX-slot, a node can transmit a single packet to its parent. During an RX-slot, a node can receive a single packet from one of its children. Let N_i denote the set of neighbors of v_i, i.e., the nodes within communication range of v_i. For a valid schedule, the nodes $v_j \in N_i$ are not allowed to transmit in the same slot and channel as v_i since the packets would collide. The goal is to choose θ as $\arg\min_\theta \sum_{p_i} \phi_i - \tau_i$, where ϕ_i is the reception time of p_i at v_0. That means, it minimizes the total end-to-end delay of the network. We call this schedule delay-minimal.

The problem description applies to TDMA systems, but also to DSME by considering only CFP slots for scheduling. It is

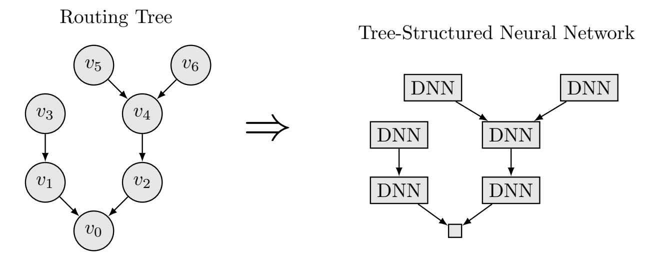

Fig. 2. Routing tree and resulting neural network architecture.

also possible to solve such problems with linear programs [5], but the scalability is not sufficient for large networks.

V. Scheduling through Reinforcement Learning

Designing a distributed scheduling algorithm that produces a globally optimized schedule is a challenging task [2]. Therefore, we employ RL to let a DNN learn the desired scheduling behavior by itself. In RL, an agent interacts with an environment and receives rewards based on how well it performs, e.g., the DNN learns which slots to schedule to reduce the end-to-end delay. The resulting DNN is executed locally at every node and the single DNNs communicate along the edges of the routing tree. As shown in Fig. 2, this forms a *tree structured neural network*, a special form of a hierarchical neural network in which individual DNNs are arranged in a tree. This tree is used for training, as further explained in V-A. Weights are shared between the DNNs, making them independent of their position in the routing tree. Additionally, this allows adding new nodes at any position in the tree because all DNNs behave in the same way. The goal is emergent behavior, in which schedules produced by simple local DNNs converge towards a globally optimized schedule.

A. DNN structure

The inputs of a DNN at v_i are the combined TX-slots of v_i's children, the TX-slots and RX-slots of its parent, the TX-slots of all $v_j \in N_i$, its queue level, δ_i, and the current slot number of the MSF. The outputs of the DNN correspond to $2K+1$ actions $a_0, \ldots, a_{2K}$, where a_i for $0 \leq i < K$ allocates k_{ij} for transmission from v_j to its parent, a_{K+i} for $0 \leq i < K$ deallocates TX-slot k_{ij} from v_j to its parent and a_{2K} corresponds to *do nothing*. A single action is selected per execution. It is sufficient to only (de)allocate TX-slots since DSME ensures that the corresponding RX-slot at the communication partner is also (de)allocated. Additionally, DSME automatically selects a free channel in the desired slot, if there is any available. The DNN uses one hidden layer with 20 neurons. At last, it must be noted that there is no communication overhead for transmitting the input parameters of the DNN to v_i. All inputs are implicitly collected during the distributed slot allocation process and are stored locally.

B. Reward function

The reward for a produced schedule is based on the resulting network delay. In theory, it can be calculated using discrete-event simulations, e.g. with OMNeT++, yielding exact delays. However, their execution time is usually too long to be applicable. Therefore, a simplified Python model was developed which calculates the delay for every packet in units of time slots. For this, a random network is generated and the DNNs of all nodes are executed. Slots are (de)allocated based on the selected actions until all DNNs choose action a_{2K}. At this time, the model calculates the average (d_{mean}) and maximum (d_{max}) end-to-end delay. The reward R is calculated as

$$R = -(\gamma_0 \cdot d_{mean} + \gamma_1 \cdot d_{max} + \gamma_2 \cdot N_{TX}), \qquad (1)$$

where N_{TX} is the total number of allocated TX-slots in the network. N_{TX} is necessary to prevent the nodes from allocating all available slots and thus limits the energy consumption of the resulting schedule to the required minimum.

VI. Discussion

As RL algorithm, Deep Double Q-Learning with experience replay and target network is used. Initial tests with the simplified Python model show good performance and for environments with $N \leq 5$, the DNN scheduler achieves minimal delay. However, increasing N to larger values results in a rapid performance decrease, especially if the number of children at a single node increases. Other RL algorithms like Monte-Carlo and Deep Deterministic Policy Gradients have been tested to solve this problem, with limited success. Therefore, we are now looking into neuroevolution to learn the network structure directly. This approach seems to reach higher scalability, although verification and comparison with TPS in openDSME and OMNeT++ are still pending. At last, the presented DNN scheduler offers flexibility in the sense that it only depends on the reward function, which can be optimized for other metrics.

VII. Conclusion

The proposed algorithm seems promising for delay-minimal scheduling in small networks. It is adaptable to different network sizes and can be trained for different objectives. Initial tests with small networks show a good performance. However, the DNNs have to be retrained if the superframe length or multi-superframe length changes. Avoiding this retraining is an obvious next step. Additionally, we are looking into increasing the overall scalability of the algorithm to large networks.

References

[1] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, 2016.
[2] F. Kauer, "Scalable Wireless Multi-Hop Networks for Industrial Applications," Ph.D. dissertation, Hamburg University of Technology, 2019.
[3] Z. Liu and I. Elhanany, "RL-MAC: a reinforcement learning based MAC protocol for wireless sensor networks," *International Journal of Sensor Networks*, vol. 1, no. 3-4, pp. 117–124, 2006.
[4] R. Mennes, M. Camelo, M. Claeys, and S. Latre, "A Neural-Network-based MF-TDMA MAC Scheduler for Collaborative Wireless Networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.
[5] F. Meyer and V. Turau, "Delay-Bounded Scheduling in IEEE 802.15. 4e DSME Using Linear Programming," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 659–666.