MASTER: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks

MASTER: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks

Laura Harms[1,2], Olaf Landsiedel[1,2]
[1]*Kiel University, Germany*
[2]*Chalmers University of Technology, Sweden*
{lah, ol}@informatik.uni-kiel.de

***Abstract*—Wireless Sensor-Actuator Networks (WSANs) are an important driver for the Industrial Internet of Things (IIoT) as they easily retrofit existing industrial infrastructure. Industrial applications require these networks to provide stable communication with high reliability and guaranteed low latency. A common way is using a central scheduler to plan transmissions and routes so that all packets are delivered before a deadline. However, existing centralized schedulers are only able to achieve high reliability in the absence of interference. This limitation lowers the feasibility of using centralized schedulers in most environments susceptible to interference.**

This paper addresses the challenge of stable, centrally scheduled communication in low-power wireless networks susceptible to interference. We introduce MASTER, a centralized scheduler and router, for IEEE 802.15.4 TSCH (Time-Slotted Channel Hopping). MASTER uses Sliding Windows, a novel transmission strategy, which builds on flow-based retransmissions instead of link-based ones. We show in our experimental evaluation that MASTER with Sliding Windows achieves routing and scheduling stability for over 24 hours with end-to-end reliability of over 99.6%. Moreover, we show that MASTER outperforms Orchestra, a state-of-the-art autonomous scheduler, in terms of latency by a factor of 8 while achieving similar reliability under a slight duty-cycle increase.

***Index Terms*—TSCH, Central Scheduling, Routing, Wireless Sensor-Actuator Networks, (Industrial) Internet of Things**

I. INTRODUCTION

For many applications in the Industrial Internet of Things (IIoT), it is essential that network traffic meets deadlines. To achieve this goal, commonly, a centralized scheduler collects information about the network topology and the wireless links. With this global knowledge, representing a major advantage over distributed solutions, the scheduler is able to compute optimal routes and transmission schedules of end-to-end communication (traffic flows). In IEEE 802.15.4, the scheduler assigns communication slots in the time and frequency domain to nodes, i.e., it employs Time-Slotted Channel Hopping (TSCH) [1]. However, due to wireless link dynamics, centralized schedulers have to account for the risk of packet losses and, therefore, usually include multiple retransmission slots for each link. These retransmission slots increase latency and reduce the available bandwidth, thus, causing an increased radio on-time.

Many recent centralized scheduling algorithms assume the availability of interference-free channels or at least a static amount of interference [2], [3]. These assumptions do not hold in many of today's environments where IEEE 802.15.4 IIoT networks co-exist with an increasingly large number of WiFi and Bluetooth networks. This coexistence results in large amounts of interference and thereby limits the stability and reliability of those centralized solutions.

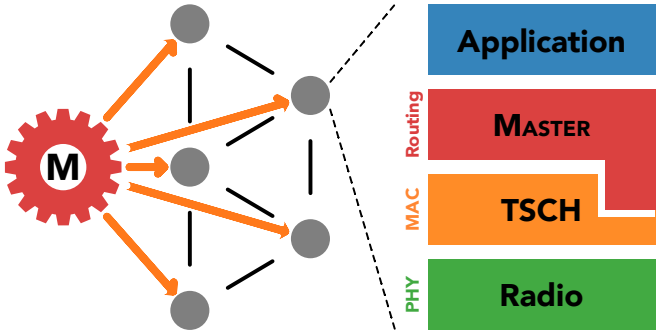


Fig. 1: MASTER consists of an external centralized scheduler (M) and a routing layer. The external scheduler performs the global routing and scheduling and pushes the computed schedule onto the network. In each node, MASTER's routing layer implements the schedule in TSCH and performs the routing during runtime.

In this paper, we introduce MASTER, a centralized scheduler designed for TSCH. It combines the traditional steps of central scheduling and routing with a novel transmission strategy which we call Sliding Windows. Our Sliding Windows algorithm introduces the flexibility needed to accomplish long-term schedule stability and communication reliability while meeting the latency requirements of industrial applications. As a result, MASTER enables long-term stable schedules and thereby eliminates the need for frequent rescheduling, a key drawback of today's central schedulers. Furthermore, we design MASTER as an open[1] and easily extendable platform to foster rapid experimentation with central scheduling policies.

Our evaluation shows that MASTER with Sliding Windows outperforms slot-based retransmission strategies of centralized schedulers. Moreover, it outperforms the low-power autonomous scheduler Orchestra [4] in terms of latency while achieving similar reliability and consuming not significantly more energy, making it particularly suitable for low-power systems. Overall, this paper makes the following contributions:

[1]Available as open-source at: https://github.com/ds-kiel/master-scheduler

- We present MASTER, an open-source, centralized router and scheduler for TSCH-based networks designed with easy extendability in mind.
- We design Sliding Windows, a transmission strategy for MASTER to increase the flexibility, stability, and reliability of centrally scheduled communications.
- We propose flow-based queues as an extension to TSCH to enable the use of central scheduling algorithms.
- We implement MASTER as part of Contiki-NG and evaluate it in environments susceptible to interference. We show the long-term stability of schedules computed by MASTER in experiments of 24 hours. These experiments result in highly reliable (>99.6%), low-latency (<4.5 slots) communications.

The remainder of this paper is organized as follows. Section II gives the necessary background information on TSCH as well as TSCH schedulers. Section III introduces the design of MASTER, and Section IV presents our testbed evaluation. Section V reviews related work, followed by the conclusion in Section VI.

II. BACKGROUND

This section gives an overview of relevant concepts on (A) Time-Slotted Channel Hopping (TSCH), (B) the ETX metric, (C) scheduling, and (D) retransmissions.

A. Time-Slotted Channel Hopping

Time-Slotted Channel Hopping (TSCH) is one of the MAC-layer protocols defined in the IEEE 802.15.4e standard [1]. TSCH uses dedicated time- and frequency-slots (TDMA and FDMA) for accessing the wireless medium. These slots are standardized to a length of 10 ms, and each slot uses one out of maximally 16 channels. TSCH continuously cycles through a hopping sequence of all active channels. Thus, it is changing the channel every slot. Assigning different frequencies to slots allows TSCH to increase the network's resilience to interference. Slots dedicated to control-information, so-called Enhanced Beacon (EB) slots, provide broadcasts which support both network formation and time synchronization, both essential for maintaining a schedule of synchronized transmissions as in TSCH.

Multiple TSCH slots are grouped into slotframes, and multiple slotframes form a TSCH schedule, see Fig. 2. Each node has a custom TSCH schedule determining its behavior in each slot. Slots are either dedicated, shared, or empty: In a dedicated slot, a node either transmits or receives. In shared slots, nodes may broadcast or receive control information, such as Enhanced Beacons. Such slots are not assigned to individual nodes and have multiple nodes contending for transmissions. To limit collisions, these slots employ the CSMA-CA back-off algorithm. If a slot is neither dedicated nor shared, it is empty, and the radio remains off to save energy.

B. Link quality metric

Link quality metrics, such as the expected transmission count, ETX [5], represent the quality of a wireless link. ETX specifies the number of transmissions expected to transmit a packet successfully over a wireless link. The ETX value is the inverse of the packet reception rate (PRR) of a link ($ETX = 1/PRR$).

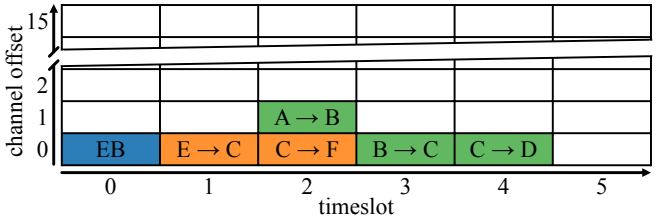


Fig. 2: Sample TSCH schedule. Slot 0 is a shared slot for sending and receiving Enhanced Beacons (EB) while slots 1-4 are unicast slots with one transmission per channel at a time. This simple schedule contains two multi-hop communication flows, highlighted in green and orange. The channel offset is added on top of the usual hopping sequence.

C. Scheduling

In the context of wireless communications, scheduling is the process of allocating resources for communications to meet all requirements such as release-time and deadline. Scheduling is an NP-hard problem, meaning, it is not optimally solvable in polynomial time (cf. [2]). Therefore, different heuristics and algorithms were developed to solve scheduling problems sufficiently well for specific scenarios.

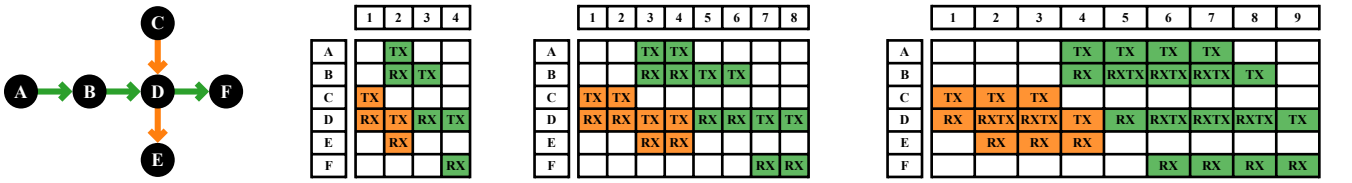
TSCH scheduling: TSCH does not specify how communications are scheduled. Therefore, scheduling TSCH communications can be performed in a centralized, distributed, or autonomous manner. In the distributed case, subsets of the network perform cooperative scheduling (cf. 6TiSCH MSF [6]). The autonomous case, used by the well-known TSCH scheduler Orchestra [4], performs an autonomous mapping of links to resources. The centralized scheduling approach provides us with global topology knowledge, and we can allocate resources using established algorithms such as Dijkstra's Shortest Path First algorithm [7].

D. Retransmissions

As wireless communication links are unreliable, transmissions are never guaranteed to be received. To increase the reliability, schedulers commonly include retransmission slots to retry a failed transmission. A common way of adding retransmissions is the duplication of single slots. This slot-based approach increases the reliability, by including multiple tries per hop. In this paper, we introduce a new, flow-based transmission strategy to increase both performance and flexibility, see Section III-B.

III. DESIGN

In this section, we present the design of MASTER, our transmission strategy Sliding Windows, and the system architecture of MASTER.



(a) Sample Topology with two intersecting flows.

(b) Baseline schedule without retransmissions.

(c) A slot-based schedule with one transmission and one retransmission slot per hop.

(d) Sliding Windows schedule with one transmission slot per hop and two (orange flow) or three (green flow) retransmission slots to be shared among the nodes of a flow.

Fig. 3: Example: One flow originates at node A to end at node F while the second one originates at node C and ends at node E.

A. *Centralized Routing and Scheduling with* MASTER

A fundamental building block of MASTER is its centralized scheduler. Its design is a three-step process to build a long-term stable, low-latency, reliable communication schedule. This process is a sequential top-down approach of (1) centralized routing, (2) applying a transmission strategy, and (3) scheduling. The input to the process is (a) a set of traffic flows specified by source, destination, periodicity, and deadline, as well as (b) the network topology with long-term link reliability statistics. The application commonly provides the set of flows, and we derive the network topology from long-term link measurements, see Section III-D5.

1) Centralized Routing: Routing is the first step in MASTER and uses the previously specified flows and network link-reliability as input. To perform the routing, MASTER constructs a directed weighted graph using an ETX-based metric (ETX^n, $n \in \mathbb{N}$, usually $n = 2$), corresponding to the link reliability statistics. A higher ETX-power favors a higher number of highly reliable links over a lower number of links with lower reliability. Using this graph, we compute the shortest end-to-end routes. As shortest path routing finds the optimal path for each flow, the flow latency selected by the routing process stays minimal. The result of our routing is an extended set of flows that consists of a source, a destination, and the intermediate hops. In MASTER, we use Dijkstra's algorithm for shortest-path routing, but our modular design allows us to plug-in any routing algorithm and metric.

2) Transmission Strategies: After computing the route for each flow, we employ a transmission strategy to ensure reliable communication over unreliable wireless links. Thus, the transmission strategy adds retransmission slots to each flow to handle failed transmissions due to link dynamics and interference. The transmission strategy extends each flow by a specific number of slots. In the case of highly reliable links in an interference-free environment, we can employ a simple transmission strategy of assigning only one slot per hop. In practice, however, we add retransmission slots according to the expected link reliability of each hop. We employ either a slot-based transmission strategy (see Section II-D) or our new approach of a flow-based transmission strategy (see Section III-B below).

3) Scheduling: After applying one of the transmission strategies, we pass the modified flows to the scheduler. The scheduler builds a communication schedule for all flows considering their periodicity.

For our application scenarios and to be comparable to Orchestra, we employ a non-deadline-based scheduling algorithm. It is especially suitable for best-effort, periodic, deadline-free systems. The algorithm is *Reverse Longest Path First (R-LPF)*, our own flavor of the *Shortest Path First (SPF)* scheduling algorithm. SPF is based on the process scheduling algorithm Shortest Job First (SJF) [8]. Contrary to starting with the shortest flow, our scheduler performs backward scheduling, starting with the end of the longest flow. This modification of the scheduling algorithm results, in our experience, in a lower number of unused slots within a flow. A lower number of unused slots corresponds with lower latency.

Fig. 3 shows a schedule for two flows generated using no retransmissions, a slot-based retransmission strategy, as well as the transmission strategy of Sliding Windows with a transmission number based on Equation (3) and a scaling factor of 1. To generate the schedule of Fig. 3d, we assume the ETX-value of each link to be between 1 and 2 ($ETX_{link} \in]1, 2[$).

Any scheduling algorithm, including deadline-based ones, can easily be implemented in MASTER. For the remainder of this paper, we use R-LPF.

B. MASTER*'s Flow-based transmission strategy*

Our flow-based transmission strategy assigns a specific number of retransmissions to a flow instead of using a per-hop basis, as done traditionally, see Section II-D. The flow-based retransmission slots allow the nodes of a flow to share these slots and use them as needed along the path, see Fig. 3d. As a result, we can increase the communication reliability while potentially using minimally more slots in the final schedule (see Node D in Fig. 3c and Fig. 3d).

With this, we divert from the traditional scheme of two active nodes to one with multiple active nodes: Traditionally, at a single time-slot, frequency, and within a localized area, only one node transmits and another one receives. Instead, we now have more than two nodes awake that either transmit or receive. Our transmission strategy has the advantage of being adaptable to network changes, e.g., due to interference.

Thus, during the journey of a packet, we can use the shared transmission slots in whichever part of the flow interference impacts communication. This adaptability is traditionally possible within distributed schedulers that can locally adapt to link changes. With Sliding Windows, we now enable such flexibility in centralized ones.

1) Window Size: The maximal number of transmission slots (TX_{max}, later denoted as $\#transmissions$) in a flow and the hop-count of the flow determine the window size which is calculated by

$$window\_size = 2 + TX_{max} - hops \quad (1)$$

This window size is the number of nodes maximally active in a slot of a flow. Moreover, it matches the maximum number of active slots of a node for a given flow. According to this relation, the window size is equal to the shared number of slots of a node for transmission or reception ($TX_{max} - hops$) plus its first and last slot allocated for reception and transmission, respectively.

In MASTER, we have two flow-based transmission policies: (1) fixed window size and (2) metric-based window size. For the first policy, we use the same window size for all flows independent of their length or link quality. For the second one, our scheduler determines the window size and number of transmissions depending on the flow's or link's ETX-values. The metric-based window size allows us to account for both the number of hops and the reliability of the individual links.

Using the link's ETX values, we can calculate the total number of transmissions of the flow with either

$$\#transmissions = n * \lceil \sum ETX_{link} \rceil, \quad n \in \mathbb{N} \quad (2)$$

or

$$\#transmissions = n * \sum \lceil ETX_{link} \rceil, \quad n \in \mathbb{N}, \quad (3)$$

including a scaling factor n. This scaling factor regulates the conservativeness of the scheduler. If we choose a scaling factor of 1 for Equation (3), the number of transmissions is equal to the one using an ETX-based, slot-based retransmission strategy (cf. Section II-D). Equation (2) uses the end-to-end ETX-value of the flow, while Equation (3) uses the ETX-values of the individual links.

Throughout the remainder of this paper, we use the following naming scheme to refer back to these equations:

$$SW- < Equation\ number > [- < scaling\ factor\ n >]$$

SW denotes it as a Sliding Windows transmission strategy. The naming scheme includes the scaling factor only if referring to a specific representation of the strategy. When referring to the general strategy, it is not included.

Please note that for long flows, i.e., with many hops such a strategy could lead to a large window, and thereby too many nodes being awake at the same point in time. Too many active nodes lead to inefficiencies, and we counterbalance it by splitting a flow into sub-flows once it exceeds a limit N. The flow-based strategy is then applied to each sub-flow individually. In MASTER, we use a threshold of $N = 10$. Thus, for example, a flow of length 11 is split into two overlapping sub-flows of length 6.

Algorithm 1 Sliding Windows transmission strategy

Input: $flow$, $graph_{ETX}$, strategy, scaling factor n
Output: $flow_{new}$ (modified version of $flow$)
1: $cost_{total} = 0$
2: **for** $i = 0$ to $length_{flow} - 1$ **do**
3: $sender_{hop} \leftarrow flow[i]$
4: $receiver_{hop} \leftarrow flow[i+1]$
5: **if** $strategy =$ "$SW-2$" **then**
6: $cost_{total} = cost_{total} + graph_{ETX}[sender_{hop}][receiver_{hop}]$
7: **else if** $strategy =$ "$SW-3$" **then**
8: $cost_{total} = cost_{total} + \lceil graph_{ETX}[sender_{hop}][receiver_{hop}] \rceil$
9: **end if**
10: **end for**
11: **if** $strategy =$ "$SW-2$" **then**
12: $cost_{total} = \lceil cost_{total} \rceil$
13: **end if**
14: $\#transmissions \leftarrow n * cost_{total}$
15: $window\_size \leftarrow 2 + \#transmissions - length_{flow}$
16: $flow_{new} \leftarrow$ list of $\#transmissions$ lists
17: **for** $i = 0$ to $length_{flow} - 1$ **do**
18: **if** $i = 0$ **then**
19: $slots \leftarrow$ list $[0\,..\,window\_size - 1]$
20: **else if** $i = (length_{flow} - 1)$ **then**
21: $slots \leftarrow$ list $[i - 1\,..\,i + window\_size - 2]$
22: **else**
23: $slots \leftarrow$ list $[i - 1\,..\,window\_size - 1]$
24: **end if**
25: **for** $slot$ in $slots$ **do**
26: extend $flow_{new}[slot]$ by $flow[i]$
27: **end for**
28: **end for**
29: **return** $flow_{new}$

2) Algorithm: In Algorithm 1, we present the algorithm for applying a flow-based transmission strategy. The algorithm takes as input a flow consisting of multiple nodes, the network's ETX graph, the strategy (SW-2 or SW-3), and the scaling factor. The algorithm starts calculating the flow's total ETX cost, as well as the flow's number of transmissions according to the given strategy (SW-2 or SW-3) and the window size according to Equation (1). From line 17 onward, the algorithm computes the active slots for each node of the flow and inserts the nodes into the respective slots of the new flow. For example, slot 6 of Fig. 3d would be represented in the new flow as a list containing the elements A, B, D, F in this order.

3) Flow-based transmissions vs. Flow Centric Policy (FCP): Recently, a paper by Brummet et al. [9] introduced a similar idea of moving from link-based to flow-based transmissions.

The main difference between Brummet's proposed Flow Centric Policy (FCP) and our Sliding Windows strategy are the rules for determining the optimal number of flow transmissions. FCP only defines fixed numbers of retransmissions with a maximum of up to 4 retransmissions for a flow. Sliding Windows, on the other hand, allows choosing the number of transmissions based on a metric, in our case, the ETX metric. Moreover, Sliding Windows allows a different number of transmissions for each flow in the same network due to its use of the ETX metric. Because Sliding Windows is based on link qualities, we argue that it offers better adaptability to a network's link characteristics during the scheduling process.

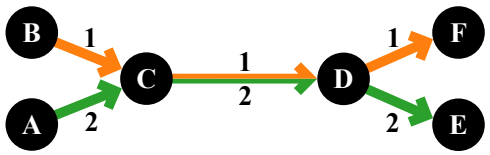


Fig. 4: Example of 2 flows sharing a common link between nodes C and D.

C. Time Synchronization

Stable time synchronization is essential for TSCH networks. It ensures that clocks do not drift apart, and nodes wake-up for transmissions and reception within the guard times specified by TSCH. MASTER achieves this by building a clock synchronization tree from the root as part of the scheduling process. Similar to the routing of the flows, a minimal spanning tree with ETX as metric and with the coordinator of the TSCH network as root is computed using Dijkstra's algorithm. This tree assigns each node a parent node for clock synchronization.

D. System Design

Next, we detail on the system architecture of MASTER. It consists of both the external scheduler and the routing layer on each node (see Fig. 1). Here we put a particular focus on the integration with TSCH and Contiki-NG [10].

1) Central Logic of MASTER*:* The central logic of MASTER consists of a centralized router and scheduler with all the functionality described above. We implement MASTER in Python to enable easy extendability and rapid experimentation of new routing, transmission, and scheduling strategies.

2) Schedule Distribution: For schedule distribution, MASTER can work together with most schedule distributors (e.g., plexi [11]), as scheduling and distributing the schedule are orthogonal. Moreover, it can also directly upload schedules via the serial port for rapid experimentation.

3) Per node routing layer: The routing layer of MASTER has multiple functions: it performs neighbor discovery (Section III-D5), implements the schedule, and adds a routing header to the communication payload to be compliant with the lower layers as well as relaying the packet to the next hop (Section III-D6). We place it in the Contiki-NG network stack above TSCH, see Fig. 1, and implement it in C.

4) Contiki-NG/TSCH Extensions: To match the requirements of MASTER and its scheduling algorithm, we extend the elements of TSCH and its implementation in Contiki-NG: (1) the packet buffer implementation and (2) the TSCH queues.

In the packet buffer, we add fields to store the flow identifier and the time to live of a transmission. With these two fields, the TSCH stack and MASTER can map incoming packets to flows and thereby follow the global schedule on each node. We extend the TSCH queue to enable a transmission order differing from the reception order at a node, e.g., the forwarding of a packet to a specific neighbor before forwarding an earlier received packet to the same neighbor. To allow this behavior, we add flow-based queues, in addition to the neighbor-based queues of TSCH. We realize the flow-based queues through the use of virtual neighbors.

Fig. 4 illustrates why neighbor-based queues as used by Contiki-NG cannot be practicably used by MASTER. If packet 2 is received by node C first, but packet 1 has an earlier deadline, packet 1 will be stuck behind packet 2 until the first is transmitted to node D. With flow-based queues, packets 1 and 2 will be added to different queues at C. Therefore, they are independent of each other and packet 1 can be forwarded first.

This new queue design increases the schedulability of the presented scheduler, which is crucial for deadline-dependent systems. It also decreases the latency in networks that are not deadline-critical by reducing congestion at bottlenecks of the network. Moreover, it allows us to use scheduling algorithms initially developed for process scheduling, a domain without these congestion problems.

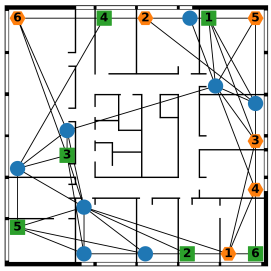
5) Neighbor Discovery and Bootstrapping: Before MASTER can build any schedule, it requires information about all links between the nodes in the network. Thus, to bootstrap and collect topology information with MASTER, we deploy a custom, topology agnostic schedule only designed for neighbor discovery. In this schedule, we use one independent transmission slot per node present in the network. This neighbor discovery schedule is similar to the sender-based operation mode of the autonomous scheduler Orchestra [4]. Each node sends a numbered broadcast in its active slot and listens in all other slots for broadcasts of other nodes in its surroundings.

Please note that this schedule only serves for bootstrapping. After deployment of the actual transmission schedule, the task of probing neighbors becomes part of the normal TSCH beaconing process. Nodes collect this information for any potential later update of the schedule.

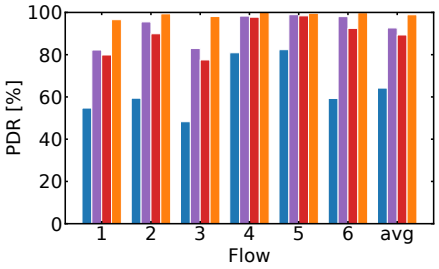
6) Header format: MASTER routes packets based on flows, and as a result, we add a custom routing header. The routing layer of MASTER adds a 7-byte routing header to each packet. This header contains a flow identifier (1 byte), a sequence number (2 bytes), the time-to-live (TTL) (2 bytes), and the earliest TSCH transmission slot (2 bytes). The header is necessary for nodes to know whether they are the receiver of the packet or a forwarder. Moreover, the header specifies, where to forward the packet to, and whether there is still time left for forwarding. In practice, our header replaces the IPv6 header which we could use instead in a system using the full IPv6 stack.

IV. EVALUATION

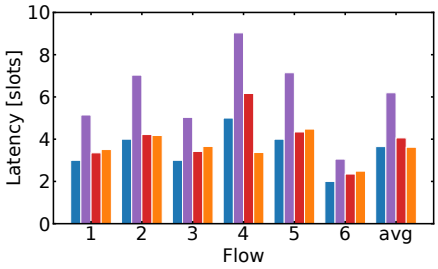
In this section, we evaluate the performance of MASTER and compare it to the state-of-the-art. We begin by evaluating our newly proposed flow-based scheduling policy and compare it to state-of-the-art scheduling policies, including a baseline strategy without retransmissions (cf. TASA [12]) and a slot-based transmission strategy (cf. AMUS [13]). Next, we compare MASTER to Orchestra, the default autonomous scheduler in Contiki-NG, which also builds on TSCH. Finally,



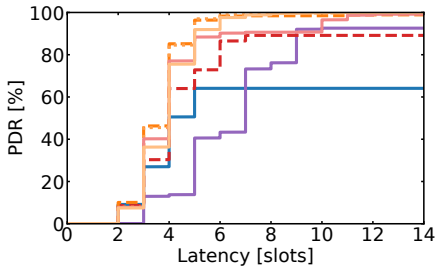
(a) 500 m^2 testbed of 20 nodes at Kiel University. Source nodes: orange hexagons; Sink nodes: green squares; Relay-only nodes: blue circles; Numbers: corresponding flow



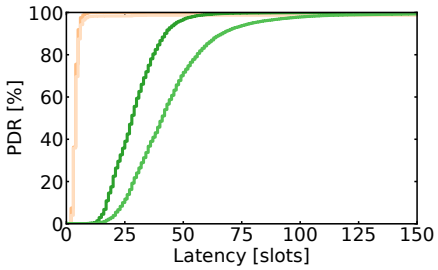
(b) Reliability of MASTER's transmission strategies: baseline, slot-based, SW-2-1 and SW-3-1.



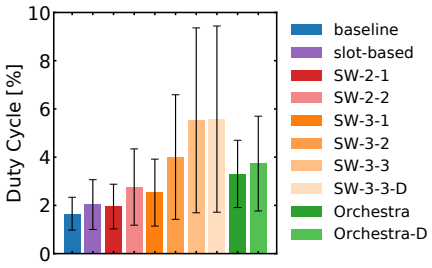
(c) Latency of MASTER's transmission strategies: baseline, slot-based, SW-2-1 and SW-3-1.



(d) Combined latency and reliability CDF of MASTER's transmission strategies.



(e) Combined latency and reliability CDF of MASTER's transmission strategy SW-3-3 and Orchestra at nighttime and daytime.



(f) Duty cycle of MASTER and Orchestra.

Fig. 5: Evaluation of MASTER's transmission strategies and comparison to Orchestra. SW-3 outperforms all other strategies reliability-wise and outperforms Orchestra latency-wise. We display the legend of figures 5b - 5f in Fig. 5f.

we evaluate MASTER's ability to compose long-term stable schedules.

A. Evaluation Setup

1) Testbed: We run on a 20 node testbed deployed in offices and student lab rooms, see Fig. 5a. It is located on the top most floor of a university building with spanning an area of 500 m^2. The testbed shares the wireless spectrum with WiFi and Bluetooth communications outside of our control. Due to this, the testbed is exposed to high levels of interference, especially during work hours.

2) Metrics, Comparison, and Duration: We evaluate our scheduler in terms of end-to-end reliability, end-to-end latency, as well as network energy consumption. We measure these metrics for different centralized scheduling approaches with and without retransmissions. Moreover, we compare our scheduler with the autonomous scheduler Orchestra [4]. These comparisons are based on 2-hour experiments for each strategy, except for the long-term stability evaluation in Section IV-E, which has a duration of 24-hours per experiment.

3) Implementation: We implement MASTER for Contiki-NG [10]. We target the Zoul Firefly platform, featuring a 32 MHz 32-bit CC2538 Cortex-M3 CPU, 32 KB of RAM, 512 KB of flash, with an IEEE 802.15.4 compatible radio.

4) Channels: Due to the high levels of interference, we use only the four channels (15, 20, 25, and 26), defined in the standard four-channel TSCH hopping sequence. Furthermore, Orchestra uses by default only these four channels as well.

5) Application Payload and Overhead: For all experiments, we include a 64-byte randomly generated data payload, a medium packet size supported by TSCH. In addition to this data payload, MASTER adds its 7-byte routing header independent of the specific scheduling policy. Orchestra, on the other hand, uses the IPv6 headers and requires additional network layer control traffic.

6) Notations: Throughout the evaluation, we use the following naming scheme: The baseline strategy without retransmissions we call *baseline*, and the slot-based retransmission strategy (as used by many state-of-art schedulers) with $\lceil ETX_{link} \rceil$ transmissions per link we label *slot-based*. The Sliding Windows strategies use the naming scheme we present in Section III-B1. Experiments performed during daytime are extended by the marker *-D*.

B. Baselines

We compare MASTER's Sliding Windows policies to three other scheduling policies. These are MASTER's baseline strategy without retransmissions, MASTER's slot-based retransmission strategy, and the autonomous scheduler Orchestra [4]. The design of the baseline strategy is based on the transmission policy used in, e.g., TASA [12], and uses one distinct slot per hop. The slot-based strategy is inspired by policies presented in

several recent publications, including AMUS [13]. Contrary to most of these, our design performs all possible retransmissions of a hop before proceeding to the next hop, which favors high reliability over low latency contrary to AMUS's approach. Moreover, to be in line with our Sliding Windows strategies, MASTER's slot-based strategy uses an ETX-based number of retransmissions per link ($\lceil ETX_{link} \rceil$). Lastly, we use Orchestra to compare our centralized routing and scheduling solution to distributedly routed and autonomously scheduled solutions to verify the adaptability of MASTER to dynamic environments predestined for distributed policies.

C. Performance of MASTER's transmission strategies

We first evaluate the performance of different transmission strategies supported by our scheduler. We compare the Sliding Windows transmission strategy with a baseline strategy without retransmissions and with the traditional slot-based retransmission strategy mentioned above. We run experiments with six scheduled flows, a number of flows used at a recent EWSN dependability competition [14]. The flows have a length of 2 to 4 hops each. Each flow has a sole source and destination node. Each source node generates a packet roughly every second with a configured time to live of one second. The length of the communication slotframes of 1 second corresponds roughly with 101 slots.

Fig. 5b shows the reliability of transmission approaches scheduled with MASTER. The transmission approaches include the baseline and slot-based strategy, as well as Sliding Windows transmission strategies SW-2-1 and SW-3-1; see Section III-B1 for notations. The latter of the two Sliding Windows strategies has the same number of transmissions per flow as the slot-based strategy.

All strategies with retransmissions clearly outperform the baseline without retransmissions, which shows the presence of interference in the used channels. The slot-based strategy reaches an average reliability of 92.7% whereas the Sliding Windows strategies reach average reliabilities of 89.3% and 98.9%, respectively. The SW-2-1 strategy has for all flows lower reliability than the slot-based strategy, but the number of scheduled slots per flow is only by one larger than the baseline number of slots, see Table I. The SW-3-1 strategy outperforms all other strategies while using no more slots per flow than the slot-based strategy. Its least reliable flow achieves a packet delivery rate (PDR) of 98.1% while the slot-based strategy drops as low as 82.2%.

We can model this superiority of SW-3-1 over SW-2-1 and over the other strategies mathematically using the probability mass function of the binomial distribution [15]:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{4}$$

This probabilistic model also explains the lower reliability of SW-2-1 compared to the slot-based strategy.

As an example, we consider a flow of three hops ($n = 3$), e.g., the green flow in Fig. 3a, with the same ETX value for each link of 1.2 ($p = \frac{5}{6}$). Thus, the number of transmissions for SW-2-1 and SW-3-1 are 4 and 6 slots, respectively. The expected PDRs for SW-2-1 and SW-3-1 are $P(X = 3) + P(X = 4) \approx 0.868$ and $P(X = 3) + P(X = 4) + P(X = 5) + P(X = 6) \approx 0.991$, respectively. Likewise, the expected PDR for the baseline, is $P(X = 3) \approx 0.579$. The slot-based strategy can be seen as 3 independent, subsequent chains of two binomial trials each ($n = 2, k \geq 1$). This results in an expected PDR of $(P(X = 1) + P(X = 2))^3 = 0.919$. These mathematical results confirm the trend we see in Fig. 5b.

TABLE I: Summary of the results plotted in Fig. 5c: Maximum latency (slots) for each flow and for flow 4 maximum number slots active in parentheses.

Flow	Baseline	Slot-Based	SW-2-1	SW-3-1
1	2	4	3	4
2	3	6	4	6
3	3	6	4	6
4	5 (3)	10 (6)	7 (4)	12 (6)
5	4	8	5	8
6	4	8	5	8

Latency-wise, both Sliding Windows strategies perform much better than the slot-based strategy. Moreover, their latency is minimally higher than the latency of a strategy without retransmissions (see Fig. 5c), which, in turn, has a high packet loss rate. It appears that SW-3-1 has a lower latency for flow 4 than the baseline. Contrary to all other flows, flow 4's schedule contains more slots than active slots throughout all strategies. Due to the flow-based approach of SW-3-1 and a large enough number of continuous active slots at the beginning of the schedule, most packets were received within a few slots, leading to a latency lower than the baseline's one. Table I shows that the maximal number of active slots is still smaller for the baseline strategy.

Fig. 5d visualizes the latency and reliability of a wider range of transmission strategies. Solid lines represent the baseline, the slot-based, the SW-2-2, and the SW-3-3 strategies. For the Sliding Windows strategies SW-2-1 and SW-3-1, the figure uses dashed lines, and for the SW-3-2 strategy, it uses a dotted line. The figure shows that the slot-based strategy is the worst latency-wise. The SW-3 Sliding Windows strategies are superior to the other Sliding Windows strategies (SW-2). The superior strategies with a scaling factor of 2 and 3, both perform well. The strategy with the higher scaling factor reaches the maximal possible reliability. Therefore, we use the Sliding Windows strategy SW-3-3 for the following comparison to Orchestra.

The duty-cycle evaluation in Fig. 5f shows a higher radio on-time for a higher number of scheduled slots. SW-3-3 has a radio on-time of up to 11.95% for a node with a lot of traffic.

D. MASTER vs. Orchestra

We now evaluate the performance of MASTER in comparison to Orchestra, the default, autonomous scheduler of TSCH in Contiki-NG. We use Orchestra as is, with a receiver-based schedule of length 7 in non-storing mode. We schedule the same six flows used before. As transmission strategy for MASTER, we use the one with the highest reliability of those

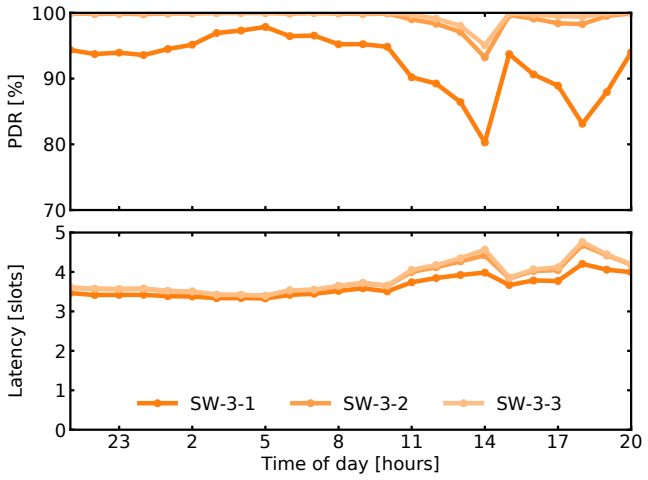


Fig. 6: Reliability and latency evaluation of Sliding Windows according to Equation (3) for all 3 scaling factors. Each value corresponds with the hour, that started at the given time. Note, that the y-axis of the PDR plot does not begin at zero.

presented above (SW-3-3). To provide detailed information on the performance, we present runs of both MASTER and Orchestra during nighttime as well as during office hours in the daytime. Fig. 5e shows the latency and reliability of the four experiments. MASTER's latency is drastically shorter than the latency of Orchestra with a mean latency of 3.9 and 4.2 slots compared to 25.9 and 40.9 slots during nighttime and daytime, respectively, while reaching similar reliability. The four rightmost columns in Fig. 5f show the duty cycle for the experiments included in this section of the evaluation. Orchestra has on average a two percentage points lower duty cycle than MASTER (3.52% vs. 5.55%) and the maximum duty cycle of a node of four percentage points lower (7.73% vs 11.95%). As each node in Orchestra is only able to use every seventh slot, the possible duty cycle is automatically lower than the one for MASTER. However, this lower duty cycle results in much higher latency, as presented above.

E. Long-term stability of MASTER

In the last part of our evaluation, we investigate MASTER's long-term stability. In Fig. 6, we present the reliability and latency of the SW-3 Sliding Windows strategies for 24 hours (Day 1, 21:00 - Day 2, 21:00) during workdays. During the night and the early morning, both SW-3-2 and SW-3-3 reach a PDR of above 99.99% and an average latency of around 3.5 slots. Between 14:00 and 15:00, the reliability drops for all strategies to 95%, 93.2%, and 80.3%, respectively, under a slight latency increase. During this time, a group of students entered the lab, leading to a drastic increase in WiFi and BLE traffic and thereby an interference level increase. Another reliability drop, mainly for SW-3-1, is visible at the end of the working day. Over the whole period of 24 hours, the average reliability of SW-3-1, SW-3-2, and SW-3-3 is 99.6%, 99.2%, and 92.5%, respectively. The high average reliability, as well as the reliability recovery after times of high interference, validates MASTER's long-term stability.

V. RELATED WORK

We first discuss centralized schedulers and algorithms, followed by a discussion of autonomous scheduling solutions.

After the introduction of TSCH, TASA [12] was one of the first central scheduling algorithms proposed. It is traffic aware, yet like other papers focusing on scheduling algorithms like C-LLF [2], it assumes the availability of interference-free channels and, therefore, does not include retransmissions. Saifullah et al. [2] and Gunatilaka et al. [3] focus in their work on the highest possible schedulability for a large amount of communications meeting deadlines but not much on the network reliability. AMUS [13] is one of the protocols for TSCH that includes slot-based retransmissions. It schedules additional resources for vulnerable links and allocates backup slots in empty cells of the scheduler. Rugamba et al. [16] build another centralized scheduler based on a path collision-aware least-laxity first scheduling algorithm by Darbandi et al. [17]. Moreover, Rugamba et al. describe a method of distributing a centrally computed schedule. The first approach of moving from slot-based retransmissions to flow-based ones is the flow-centric policy (FCP) [9]. The authors present a dynamic approach of retransmissions not fixed to specific links. This approach is similar to the transmission strategy of Sliding Windows presented in this paper. We discuss the differences between the two in Section III-B3.

Besides the advances regarding scheduling, Wu et al. [18] present advances in the field of centralized routing in combination with central scheduling. The authors present a conflict aware real-time routing approach, that is aware of scheduling decisions and the possible conflicts of routed paths. Li et al. [19] take a different, asymmetric approach in routing by applying different routing strategies for different communications in one network.

Related to these central scheduling and routing approaches, are systems focusing on network softwarization. plexi [11] is a framework exposing TSCH network resources through a web interface and allowing the rescheduling of communications. Similarly, Baddeley et al. [20] and Galluccio et al. [21] present SDN solutions for Wireless Sensor Networks for network monitoring and reconfiguration. These SDN solutions are conceptually in line with central schedulers calculating schedules externally. Moreover, a combination of our work with SDN solutions is imaginable.

Next to the centralized approaches, a significant focus of recent work is on autonomous scheduling, a concept introduced by Orchestra [4]. Orchestra, as well as Alice [22] and DiGS [23] are autonomous solutions for TSCH, as they do not require neither any central infrastructure nor the exchange of data to build a schedule and achieve high reliabilities of 99.999%. However, autonomous schedulers are not able to achieve this reliability with latency guarantees necessary for many industrial applications as they have no knowledge on the underlying topology.

VI. Conclusion

This paper introduces MASTER, a central scheduling solution for TSCH networks. MASTER introduces a novel Sliding Windows transmission strategy and achieves high reliability independent of knowing the optimal amount of retransmissions per link. Instead, it schedules a number of retransmissions for a flow that can be used at all links of a flow where necessary. The key idea is enabling centralized schedulers to adapt to interference changes without the need for rescheduling while keeping the lowest possible latency. Thus, eliminating a significant overhead of traditional central schedulers.

We implement MASTER in Contiki-NG and evaluate it extensively on a testbed in an environment susceptible to interference. We demonstrate MASTER's practicality and ability to keep stability for over 24 hours and achieve latencies much smaller than Orchestra while achieving similar reliability.

As part of future work, we plan to investigate the challenges of neighbor data collection and schedule distribution to provide a comprehensive central scheduling solution. Moreover, we are planning to evaluate the use of centralized schedulers in harsh wireless environments, such as the ones used in the EWSN dependability competitions [14].

References

[1] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," IEEE, Tech. Rep. [Online]. Available: http://ieeexplore.ieee.org/document/6185525/

[2] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *IEEE RTSS*, 2010, pp. 150–159.

[3] D. Gunatilaka and C. Lu, "Conservative Channel Reuse in Real-Time Industrial Wireless Sensor-Actuator Networks," in *IEEE ICDCS*, 2018, pp. 344–353.

[4] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *ACMSenSys*, 2015, pp. 337–350.

[5] D. S. J. De Couto, "High-Throughput Routing for Multi-Hop Wireless Networks," PhD thesis, MIT, 2004. [Online]. Available: https://pdos.lcs.mit.edu/papers/grid:decouto-phd/thesis.pdf

[6] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," Internet draft, Tech. Rep., Mar. 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6tisch-msf-14

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 24.3: Dijkstra's algorithm," in *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, 2001, pp. 595–601.

[8] A. S. Tanenbaum and H. Bos, "Section 2.4.2: Scheduling in Batch Systems - Shortest Job First," in *Modern Operating Systems*, 4th ed. Pearson Education, Inc., 2015, pp. 157–158.

[9] R. Brummet, D. Gunatilaka, D. Vyas, O. Chipara, and C. Lu, "A Flexible Retransmission Policy for Industrial Wireless Sensor Actuator Networks," in *IEEE ICII*, 2018, pp. 79–88.

[10] "Contiki-NG: The OS for Next Generation IoT Devices." [Online]. Available: http://www.contiki-ng.org/

[11] G. Exarchakos, I. Oztelcan, D. Sarakiotis, and A. Liotta, "plexi: Adaptive re-scheduling web-service of time synchronized low-power wireless networks," *Journal of Network and Computer Applications*, vol. 81, pp. 62–73, Mar. 2017.

[12] M. Palattella, N. Accettura, M. Dohler, L. Grieco, and G. Boggia, "Traffic-Aware Time-Critical Scheduling in Heavily Duty-Cycled IEEE 802.15.4e for an Industrial IoT," in *IEEE PIMRC*, 2012, pp. 327–332.

[13] 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 *IEEE WCNC*, 2016, pp. 1–6.

[14] C. A. Boano and M. Schuß, "EWSN 2019 Dependability Competition Logistics Information, rev. 1," Jan. 2018. [Online]. Available: https://iti-testbed.tugraz.at/fileupload/static/fileupload/EWSN2019_DC_Logistics_1.pdf

[15] L. Råde and B. Westergren, *Mathematics Handbook for Science and Engineering*. Lund: Studentlitteratur AB, 2004.

[16] J. P. G. Rugamba, D. L. Mai, and M. K. Kim, "Implementation of a Centralized Scheduling Algorithm for IEEE 802.15.4e TSCH," in *Intelligent Computing Methodologies*, D.-S. Huang, Z.-K. Huang, and A. Hussain, Eds. Springer Int. Pub., 2019, vol. 11645, pp. 118–129.

[17] A. Darbandi and M. K. Kim, "Path Collision-aware Real-time Link Scheduling for TSCH Wireless Networks," *KSII Transactions on Internet & Information Systems*, vol. 13, no. 9, 2019.

[18] C. Wu, D. Gunatilaka, M. Sha, and C. Lu, "Real-Time Wireless Routing for Industrial Internet of Things," in *IEEE/ACM IoTDI*, 2018, pp. 261–266.

[19] B. Li, Y. Ma, T. Westenbroek, C. Wu, H. Gonzalez, and C. Lu, "Wireless Routing and Control: A Cyber-Physical Case Study," in *ACM/IEEE ICCPS*, 2016, pp. 1–10.

[20] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for Low-Power IoT Networks," in *IEEE NetSoft*, 2018, pp. 71–79.

[21] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *IEEE INFOCOM*, 2015, pp. 513–521.

[22] S. Kim, H.-S. Kim, and C. Kim, "ALICE: autonomous link-based cell scheduling for TSCH," in *IEEE/ACM IPSN*, 2019, pp. 121–132.

[23] J. Shi, M. Sha, and Z. Yang, "DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks," in *IEEE ICDCS*, 2018, pp. 354–364.