Scheduling High-Rate Unpredictable Traffic in IEEE 802.15.4 TSCH Networks

Atis Elsts, Xenofon Fafoutis, James Pope, George Oikonomou, Robert Piechocki and Ian Craddock
Department of Electrical and Electronic Engineering,
University of Bristol, UK

***Abstract*—The upcoming Internet of Things (IoT) applications include real-time human activity monitoring with wearable sensors. Compared to the traditional environmental sensing with low-power wireless nodes, these new applications generate a constant stream of a much higher rate. Nevertheless, the wearable devices remain battery powered and therefore restricted to low-power wireless standards such as IEEE 802.15.4 or Bluetooth Low Energy (BLE). Our work tackles the problem of building a reliable autonomous schedule for forwarding this kind of dynamic data in IEEE 802.15.4 TSCH networks. Due to the *a priori* unpredictability of these data source locations, the quality of the wireless links, and the routing topology of the forwarding network, it is wasteful to reserve the number of slots required for the worst-case scenario; under conditions of high expected datarate, it is downright impossible. The solution we propose is a hybrid approach where dedicated TSCH cells and shared TSCH slots coexist in the same schedule. We show that under realistic assumptions of wireless link diversity, adding shared slots to a TSCH schedule increases the overall packet delivery rate and the fairness of the system.**

***Keywords*-Time slotted channel hopping, scheduling, Internet of Things.**

I. Introduction

The recently published IEEE 802.15.4-2015 standard [1] introduces several new MAC (Medium Access Control) modes of operation. Among others, it standardizes the Time-Slotted Channel Hopping (TSCH) protocol, a TDMA (Time Division Multiple Access) MAC layer. TSCH has attracted significant attention from the research community because it promises more reliable and predictable wireless networking for the Internet of Things (IoT). Because of channel hopping at the MAC layer, TSCH is able to withstand narrow-band interference and multipath fading. Additionally, its time-slotted nature means that the protocol is comparatively easier to predict, in terms of energy consumption and delay, as long as an appropriate communication schedule is provided. However, constructing efficient schedules for applications using TSCH networks is still an open research problem.

In this paper we focus on increasing the reliability of data collection in TSCH networks characterized by *high datarate* and *unpredictable* traffic. By *high datarate* we mean conditions where the network sink's packet reception capacity is close to being maximally utilized. By *unpredictable* we mean the combination of several aspects: (1) hard-to-predict data rates and locations of mobile data sources; (2) hard-to-predict wireless link dynamics; (3) hard-to-predict routing path changes. The timescale of these changes, in particular (1) and (2), is on the order of seconds or even milliseconds; this does not play well with reactively adaptive scheduling mechanisms, as rapid-enough adaptations become impractical or downright impossible. Instead, proactive overallocation of scheduled TSCH timeslots is necessary. However, such an overallocation significantly reduces the total schedule capacity, and in normal operating conditions it causes many reserved slots to be left unused.

The *main contribution* of this work is a method to increase the number of successful packet transmissions within a TSCH slotframe, and therefore the network's total throughput. We achieve that by *sharing* some of the transmission slots between different nodes. Essentially, this is hybrid scheme where slotted-Aloha (contention-based slots) and dedicated unicast timeslots (contention-free slots) coexist in the same TSCH slotframe. We show that the throughput is increased if either: (a) there are significant differences in the quality of links between multiple upstream nodes and the same upstream node. Numerous real-world studies [2] [3] [4] confirm that this is fair assumption in low-power wireless networks; links spatially separated by less than a meter often show completely different performance. Or, (b) the nodes have unequal requirements in terms of how many messages they need to transmit. This in particular characterizes multihop topologies, where nodes with more children need to forward more data towards the root.

For an illustrative example, consider the Fig. 1. Here, three out of four nodes have good link-layer Packet Reception Rate (PRR), $PRR = 0.8$, while the node A has an average quality ($PRR = 0.4$) link to the gateway. To deliver 100 % of packets, A has to retransmit each packet 2.5 times on the average. In a scenario where each of the nodes generates 10 packets per second, approximately 12.5 slots per second are sufficient for B, C, and D; this number is slightly higher than $10 \times ETX = 12.5$ because of finite queue sizes. On the other hand, A requires at least 25 dedicated slots. Since there are only 100 TSCH slots per second, a static TSCH schedule that is symmetrical (*i.e.*, makes no a priori assumptions about which links may be bad) would exhaust all its capacity just for one-directional communication. Our solution is to reserve some slots for common usage. For example, if 16 slots are reserved for each node and 16 slots are shared between them, A has access to the total of 32 slots, which is both sufficient to deliver all its data and leaves 20 slots per

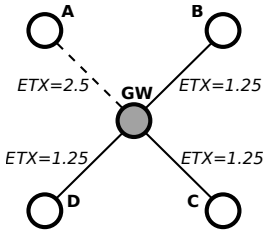


Figure 1: An example TSCH forwarding network topology. Links from B, C and D all have good quality ($PRR = 0.8$), link from A has average ($PRR = 0.4$), therefore A has to use more retransmissions to deliver an equal number of packets.

second usable for other traffic (*e.g.*, broadcast and gateway-to-node traffic). We rely on an autonomous scheduling approach that requires no run-time communication between the nodes besides the standard TSCH control messages and RPL routing messages. The schedule is first statically constructed at compile-time; subsequently, at the run-time, nodes autonomously decide which and how many TSCH links to use, depending on their routing state. Using a queue-size dependent contention algorithm for the shared slots, the nodes with lower link quality *or* with more data to send are able to make more transmissions within each slotframe.

The number of children nodes of a node affects its cumulative data rate. Our *secondary contribution* is a mechanism that allows nodes to autonomously reappropriate dedicated slots belonging to other nodes, as long as collision-free operation is guaranteed. The nodes do this by using their RPL routing state; in particular, slots belonging to indirect children nodes can be safely reused when transmitting data upstream. Our results shows that this approach leads to better results than simple slot sharing in multihop topologies.

This work is motivated by a real-world use case (Section III). At the same time, we believe that the assumptions in this work are typical enough for many other applications.

We start with discussing some background information and related work (Section II) and the motivating use case (Section III). Subsequently we present analytic modeling results (Section IV) as a motivation for the the shared slot approach. The design of the system is described in Section V. To assess the performance of the system (Section VI), we first evaluate the slot-sharing approach numerically, then implement the scheduling algorithm in the Contiki OS and evaluate the implementation using simulations in Cooja, the Contiki simulator. The outcome shows a good match between the numerical and full-scale results, and reduction in packet loss rate up to 3.5–4 times when the shared slot approach is used.

II. Background and related work

Time in TSCH networks is globally synchronized and is divided in *timeslots*. Typically, each timeslot is 10 ms long, and supports the transmission / reception of a single packet along with its acknowledgment. Timeslots are grouped in *slotframes*. A TSCH *schedule* consists of one or more periodically repeating slotframes.

The TSCH specification [1] supports both contention-based (shared) slots and contention-free (dedicated) slots. A schedule with only shared slots provides slotted-Aloha behavior with maximal flexibility, as no assumptions about the network's topology and packet generation rates are needed. In contrast, a schedule of dedicated slots is more reliable, but cannot be effectively constructed without predicting the traffic flows and link qualities in the network.

The 6tisch IETF working group [5] is an ongoing effort to tackle the challenges of running IPv6 traffic on top of low-power TSCH networks. One of the outcomes is a minimal schedule proposal for TSCH networks [6]. This minimal schedule consists of just one active shared slot per slotframe. Their main goals are interoperability and basic functionality in the absence of more complicated schedules, as well as support for event-driven (unpredictable) traffic patterns; the 6tisch minimal schedule does not make any assumptions about the network topology or the traffic patterns.

Orchestra [7] is an autonomous scheduling mechanism where each node independently decides its schedule based on its RPL routing state. Time slots in Orchestra can be either shared (*e.g.*, between the children of a node) or dedicated. Orchestra compared to the 6tisch minimal schedule reduces the number of packet collisions and leads to an order-of-magnitude in the number of undelivered end-to-end packets [7]. The main difference from our work is that Orchestra is not targeted towards high-datarate networks. In Orchestra, each slotframe only has up to one active slot for each pair of nodes, and the authors do not tackle the problem of congestion points appearing in multihop networks.

Juc *et al.* [8] compare two of the new MAC schemes introduced in IEEE 802.15.4 standard: DSME (Deterministic and Synchronous Multichannel Extension) and TSCH. DSME is an extension on top of beacon-enabled IEEE 802.15.4 networks where guaranteed timeslots are present. DSME supports both contention-free and a contented slots inside each slotframe. Juc *et al.* conclude that DSME shows higher throughput for high-duty cycle applications. However, this conclusion holds only in case short packets are used. If Tx-ing, processing, and acknowledging a packet takes up the whole duration of a TSCH slot (10 ms), the network is already operating at its theoretical capacity. Our experiences [9] with porting TSCH to CC2650 verify this observation. Older platforms such as TelosB and Zolertia Z1 are forced to use even longer TSCH timeslots (15 ms).

A plethora of research work discusses designing centralized [10] [11] or distributed [12] schedules for TSCH. However, there is lack of implementation-tested publicly available work, not the least because of run-time complexity of implementing these schemes.

Hashimoto *et al.* [13] in particular investigate sharing TSCH slots. They conclude that sharing slots leads to

shorter slotframe sizes, therefore to reduction of delay and improvements of throughput. However, in their work the slots are shared between *flows* going through the same node, not shared between nodes.

III. The SPHERE Use Case

SPHERE (Sensor Platform for HEalthcare in Residential Environment) is an interdisciplinary research collaboration with the objective to develop a multipurpose, multi-modal sensor platform for monitoring people's health inside their homes [14]. Untypically for academic projects, in 2017 the SPHERE platform is going to be deployed in real people's homes (100-home study), therefore it has to satisfy many real-world requirements.

The SPHERE project is using three sensing modalities to capture health-related data [14]: (1) a low-power environmental sensor network, including light, humidity, temperature, presence, and noise sensors (E devices in Fig. 2); (2) a network of low-power wearable devices equipped with accelerometer sensors, one per home resident (W devices in Fig. 2); (3) a video sensor network for recognition of activities through video analysis.

The data from wearable and environmental sensors is collected using a backbone network consisting of a number of mains-powered *forwarding nodes* deployed through the house (G devices in Fig. 2). These nodes have dual radios, running both Bluetooth Low Energy (BLE) and 6LoWPAN over IEEE 802.15.4 TSCH stacks at the same time.

The wearable devices continuously measure 3-axis acceleration with 25 Hz frequency [15]. At least 20 Hz frequency acceleration measurements are required to be collected on the server for accurate activity detection of the human participants [16], which is one of the project's main goals. The measurements are continuously broadcast using BLE advertising mode. We use broadcast in order to (1) make use of spatial diversity of the forwarding nodes for more reliable data collection and (2) implement RSSI-based room-level localization of the human participants.

The wearable advertisements are picked up and forwarded by the TSCH network node towards the network's sink. A single TSCH packet contains 12 readings of the three-axis accelerometer together with the overhead of metadata and packet headers. This leads to packet generation rate of 2.1 TSCH packets per second per single wearable device. However, multiple forwarding nodes are expected to pick up each wearable advertisement on the average. The location of the wearable devices is highly unpredictable, as it depends on both human movement and wireless link quality within the house. Therefore the data rate on the individual forwarding nodes cannot be predicted before deploying the network; furthermore, the rate is expected to be highly dynamic, therefore designing on-the-fly schedule allocation techniques for forwarding the wearable data is nontrivial to say the least.

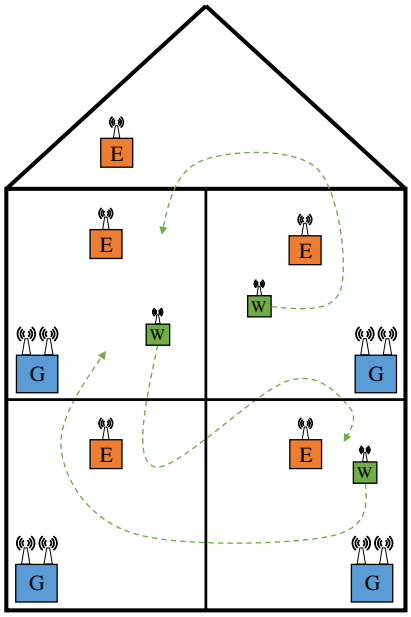


Figure 2: An example of the SPHERE platform for residential monitoring. The mains-powered forwarding nodes (G) form a backbone TSCH network and collect data from the battery-powered environmental sensor nodes (E). In addition, they are equipped with an additional BLE radio for collecting the data broadcast by the mobile wearable sensors (W).

The system is required to support up to 4 wearable devices. Assuming the worst-case scenario of all forwarding nodes picking up all wearable advertisements, each forwarding node is going to generate 8.3 packets per second. To make matters worse, TSCH timeslot allocation should account for link-layer retransmissions. The level of this overallocation depends on the quality of the channel, which in an indoor environment is notoriously dynamic, due to multipath propagation, channel fading and body shadowing. Assuming that $PRR = 0.5$ is the worst level of link quality the system needs to survive, 16.6 TSCH slots per second are required for data forwarding from each forwarding node. Since there are only 100 TSCH slots per second, that leaves us with very limited maximal number of forwarding nodes even assuming a single hop network: only up to $\lfloor 100/16.6 \rfloor = 6$ nodes. A multihop network makes things worse, as a node with n children is expected to generate $n + 1$ times more data in the worst case compared to a node with no children. On the other hand, the average long-term link quality of a properly deployed indoor wireless network is expected to be closer to $PRR = 1.0$ than to $PRR = 0.5$, especially as our system uses adaptive techniques to avoid heavily interfered IEEE 802.15.4 channels [17]. If the overallocated slots could be reduced, up to two times fewer total slots would be required for each forwarding node. The practical objective of the work behind this paper is to exploit this fact in order to support a higher number of forwarding nodes without sacrificing reliability.

IV. System model

In high rate scenarios with multiple end nodes, a slotted-Aloha TSCH schedule is expected to perform very poorly, primarily due to the very high collision probability. On the other hand, a sufficient number of dedicated slots for packet retransmissions severely restrict the maximal number of nodes supported in a star (sub)network.

Consider a single-hop TSCH network with two end nodes, A and B, and a single sink node. The end nodes have a datarate of r_A and r_B respectively, expressed in packets per slotframe. For simplicity, let us disregard channel hopping, and assume that the link from the end nodes to the sink is characterized by a link-layer PRR, p_A and p_B respectively. We model retransmissions as a sequence of independent Bernoulli trials with the same probability of failure (*i.e.* PER) for each trial. Furthermore, we assume infinite queues. The total expected number of transmissions for node A, R_A, can therefore be calculated as

$$R_A = r_A \sum_{n=1}^{\infty} n p_A (1-p_A)^{n-1} = \frac{r_A}{p_A} \ . \tag{1}$$

The total expected number of transmissions for node B, R_B is calculated similarly. R_A and R_B are also expressed in packets per slotframe. N_F defines the number of slots in a slotframe. $N_S \leq N_F$ defines the number of shared slots within a slotframe. In addition, each end node has $N_D = \frac{1}{2}(N_F - N_S)$ dedicated contention-free slots.

In each slotframe, each end node first uses its dedicated slots for up to N_D packets. It then attempts to forward the remaining traffic, if any, via the shared slots. We denote the excess traffic as $C_A = R_A - N_D$ and $C_B = R_B - N_D$ respectively. If there are no shared slots ($N_S = 0$), the excess traffic is lost. Otherwise, it is transmitted through the shared slots, with a probability of collision. We assume that each shared slot is selected with a probability $\frac{C_A}{N_S}$ and $\frac{C_B}{N_S}$ respectively, and we model the total number of collisions (K) in a slotframe as the cumulative probability of A and B selecting the same slot:

$$K = N_S \frac{C_A}{N_S} \frac{C_B}{N_S} = \frac{C_A C_B}{N_S} \ . \tag{2}$$

The end-to-end packet delivery rate (PDR) of node A, denoted as $\texttt{PDR}_A$, can be then estimated as follows:

$$\texttt{PDR}_A = \begin{cases} 1 - \frac{C_A}{R_A} & N_S = 0 \\ 1 - \frac{K}{R_A} & N_S > 0 \text{ and } C_A \leq N_S \\ 1 - \frac{K + C_A - N_S}{R_A} & N_S > 0 \text{ and } C_A > N_S \ . \end{cases} \tag{3}$$

The PDR of node B, denoted as $\texttt{PDR}_B$, is estimated similarly. The overall performance of the TSCH network is optimized at the number of shared timeslots (N_S) that maximizes the average PDR. Fig. 3 shows numerical results for a scenario with asymmetrical link qualities, *i.e.* $p_A = 0.95$ and $p_B = 0.55$. The traffic rate is the same for both nodes:

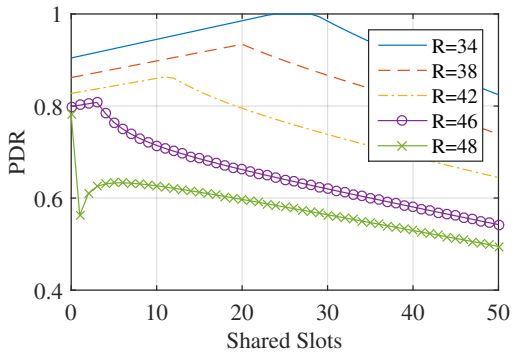


Figure 3: Slot sharing improves the reliability of links with asymmetrical quality. The optimum number of shared slots decreases as the overall traffic increases.

$R = r_A = r_B$. The slotframe size is $N_F = 100$ slots. The x-axis is the number of shared slots, N_S, and the y-axis is the average PDR of A and B. Using a contention-free schedule, *i.e.* $N_S = 0$, in this asymmetrical instance is not ideal: there are unused slots for node A, whilst there are not enough slots for node B. On the other hand, slot sharing improves the performance. We can also observe that the optimum number of shared slots decreases as the traffic increases, until a certain level where a contention-free schedule is optimal.

V. System architecture

Schedule design. The following kinds of cells must be supported: (a) cells for broadcast traffic; (b) cells for transmission from forwarding nodes to the gateway; (c) cells for transmissions between the forwarding nodes; (d) other cells, *e.g.*, for communication with leaf nodes. The schedule is constructed as follows. At the start of the slotframe, there is a single broadcast slot (category a). Subsequently, there is a repeating block of slots for traffic categories b and c. In a network with N forwarding nodes, the block consists of N slots (Fig. 4) for transmission from each node, followed by zero or more shared slots. Finally, a few slots at the end of slotframe are reserved for communication with leaf nodes (category d; not further discussed in this paper, as that is low data rate traffic). We assume that the number of forwarding nodes in the network is known at compile time. In these conditions, it is possible to map the MAC address of each node to a unique channel offset, subsequently called *the node's channel offset*. In practice, a lookup table or a hash function can be used to perform this mapping. As long as the number of channels used in the network is greater or equal that the number of forwarding nodes, the communication with different nodes takes place on different channels.

Runtime behavior. A forwarding node wakes up at the start of every TSCH time slot. Subsequently, its actions are:

- If the slot is a broadcast slot, use channel offset 0:
  - if broadcast packets are queued, send out one of them;
  - otherwise, listen for packets.

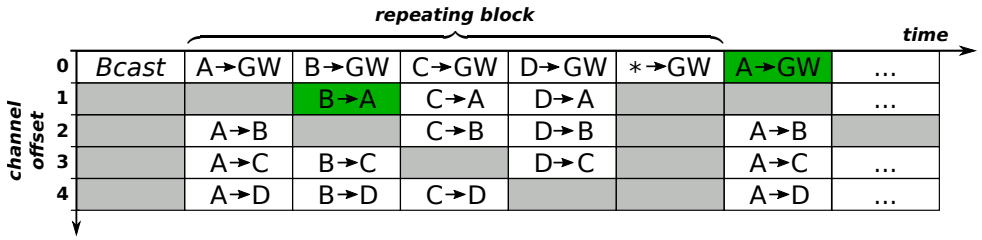


Figure 4: A fragment of the slotframe for networks with 4 forwarding nodes, such as in Fig. 1 and Fig. 5. GW denotes the gateway node, A, B, C, D denote forwarding nodes. *Bcast* marks a timeslot reserved for broadcast. $* \rightarrow GW$ denotes a shared slot, in which all nodes directly connected to the gateway are allowed to transmit. The two green cells in the schedule mark one way how to forward a packet from B to the gateway (GW), assuming network as in Fig. 5a.

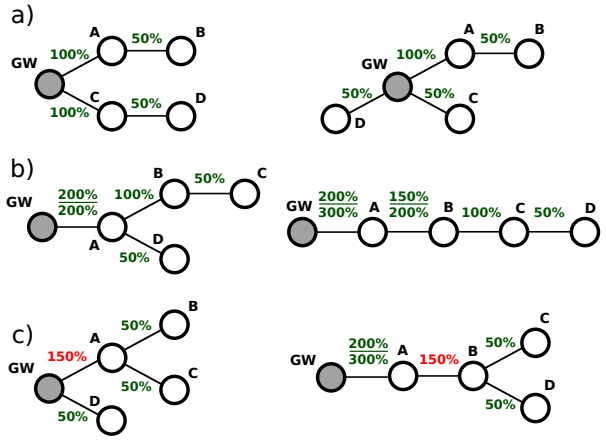


Figure 5: Schedule capacity usage in different topologies assuming worst-case data generation rate on each node and 100 % cell overallocation. (a) A topology supported naturally. **(b)** A topology supported by applying the cell reuse technique. Node A is able to use cells scheduled for $C \rightarrow GW$ for its own data forwarding to GW in a collision-free way, since A has C as an indirect child in its routing tables. **(c)** A topology not supported assuming the worst-case data rate.

- Otherwise, if the slot has a cell that denotes communication with a leaf node, perform the required transmission with the scheduled channel offset.
- Otherwise, if the slot contains cells where the node is marked as sender: select a neighbor node that has nonempty packet queue, if such any exists. Use the selected neighbor node's channel offset to transmit a packet from the neighbor's queue.
- Otherwise, if the slot is a shared Tx slot and the node is directly connected to the gateway, randomly choose whether to transmit a packet or sleep, with probability depending on the number of packets in queue.
- Otherwise, listen for packets using node's channel offset.

The algorithm is similar on the gateway node, except that: (1) the GW node always listens during shared slots and (2) it reuses network's broadcast slots to transmit unicast packets.

Contention. Since the quality of real-world wireless links varies significantly, unpredictably, and rapidly, significant slot overallocation is required to account for retransmissions. Under normal channel conditions the overallocated slots remain idle. Wireless link studies [2] [3] [4] show that even though wireless link quality is occasionally correlated (*e.g.*, because of regional external interference), some spatial diversity is always present as well. Given this spatial diversity assumption, we can conclude that it is very unlikely that all links are going to suffer from bad performance at the same time. Therefore the overallocation factor can be reduced by introducing shared slots. To minimize the number of collisions in these shared slots, only nodes with bad quality links or excessive data to forward should make use of those links. It's easy to see that if contention points in the network exist at all, the gateway node is always one of them, as it receives more data than any of the other nodes. Therefore in our scheme the shared slots are reserved for transmissions specifically to the gateway node. In a shared slot, the probability that a node transmits a packet is selected randomly, as a function of that node's queue size q: $P_{\text{Tx}} \equiv f(q)$. We found that both linear dependency and squared dependency show good results. Further in the paper, f is selected to be square of the queue size: $f(q) = \frac{q^2}{S}$, where S is the number of shared slots per slotframe.

Multihop. Additional overallocation is required because of multihop topologies. If the maximal data rate of a single node is K, then the maximal cumulative data rate of a node forwarding data from N children is $N \times (K + 1)$. Consequently, static scheduling in multihop topologies with this method either requires massive overallocation for the worst-case number of children, or leads to some topologies where the schedule cannot satisfy PDR requirements because of contention. To increase the throughput, we implement *slot reuse* on the forwarding node, where slots "belonging" to its indirect children are appropriated by the node for traffic forwarding upstream. For example, assuming line topology as in 5b, the node A is free to reuse cells scheduled for $C \rightarrow GW$ and $D \rightarrow GW$ to send its data to the gateway, while node B can use the cell $D \rightarrow A$ to send packets to A. Note that A is not able to use cells $B \rightarrow GW$, because during that timeslot A is already busy with listening for packets coming from B.

Ensuring fairness. In a transmission slot, a node can select any of its neighbors with nonempty packet queue as the receiver. It is clear that always preferring neighbors in the

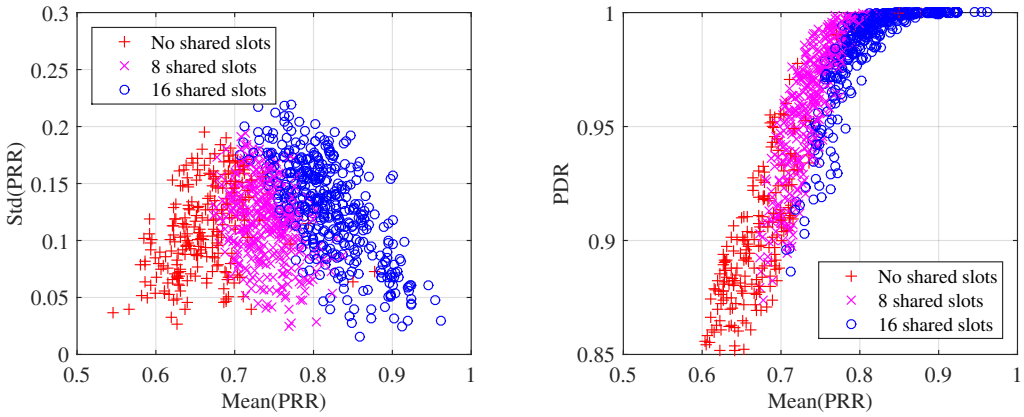


Figure 6: Scatter plots showing the schedule with the best performance for 1000 simulation experiments with randomly-selected link qualities for each link. Each scenario is simulated with a contention-free schedule (no shared slots) and with two hybrid schedules (8 shared slots and 16 shared slots) and the best configuration is plotted. The optimum number of shared slots increases as the average link quality increases, but also as the standard deviation of the quality of the links of the TSCH network increases.

same order can lead to starvation of the less-prioritized ones, *i.e.*, situations where the packets in the neighbors' queues are not sent out before they timeout. We perform the selection in a round robin fashion, using the current timeslot as the input. Let us denote with N the number of nodes and with $node\_id$ the local node's channel offset. This algorithm prioritizes all neighbors equally as long as the slotframe length and N are co-prime numbers:

for $i = 0;\quad i < N;\quad i = i + 1$ **do**
  $neighbor\_id \leftarrow (i + timeslot)$ **mod** N
  if $neighbor\_id \neq node\_id$ **then** ▷ Not the local node
    if $packet\_queue[neighbor\_id] \neq \emptyset$ **then** ▷ Has packets
      return $packet\_queue[neighbor\_id].pop()$
    end if
  end if
end for
return $NULL$

VI. Experimental setup and results

A. Numerical simulations

In this section, we use numerical simulations to validate the analytic observations of Section IV and evaluate the benefits of TSCH slot sharing in multiple diverse scenarios that are likely to be encountered in real-life deployments, such as the SPHERE 100-home deployment. To this end, we have implemented a discrete-time Python simulator[1], which can simulate different TSCH networks and estimate the average PDR under various schedules. Using this simulator, we compare our solution against a contention-free schedule.

More specifically, a single-hop TSCH network is simulated as follows. When a contention-free schedule is selected, the available slots are divided equally to all the end nodes. When a hybrid schedule with shared slots is simulated, the number of shared slots are reserved, whilst the remaining slots are divided equally amongst the end nodes. Each node forwards its traffic primarily through its dedicated slots, using the shared slots for excess traffic. In the simulations that follow we assume that packets are dropped after a maximum of 8 transmissions. In addition, we assume a queue size of 8 packets. Moreover, we assume a slotframe size of 99 slots. We reserve 19 of these slots for broadcasting and other traffic. The remaining 80 of the these slots are used by 4 end nodes that form a star network, similar to Fig. 1. Each simulation estimates the average PDR of the 4 end nodes for a period of 600 seconds.

In the first series of simulations, we compare three schedules: (i) 0 shared slots, (ii) 8 shared slots, and (iii) 16 shared slots. We simulate 1000 instances of the topology, considering that each end node forwards 14 packets per slotframe. In each instance, the quality of the links is randomly selected in [0.5-1.0]. For each configuration, the PDR of each of the thee schedules is estimated separately, and the best schedule is plotted in the scatter plots in Fig. 6. In Fig. 6a, the x-axis is the average PRR of the links and the y-axis corresponds to their standard deviation. In Fig. 6b, the x-axis is the average PRR of the links and the y-axis corresponds to the resulting average PDR.

The results demonstrate that slot sharing is beneficial in many scenarios. Moreover, the simulations verify the key observation of Section IV. Indeed, as the average quality of the links decreases and the overall traffic increases due to retransmissions, the best performance is achieved with a smaller number of shared slots. In particular, when the average PRR is less than 0.7, the contention-free schedule yields the optimum performance. Similarly, the hybrid schedule with 8 shared slots dominates in networks with an average PRR of $0.7 - 0.8$; and the hybrid schedule with

[1]Available in https://github.com/irc-sphere/tsch-simulator.

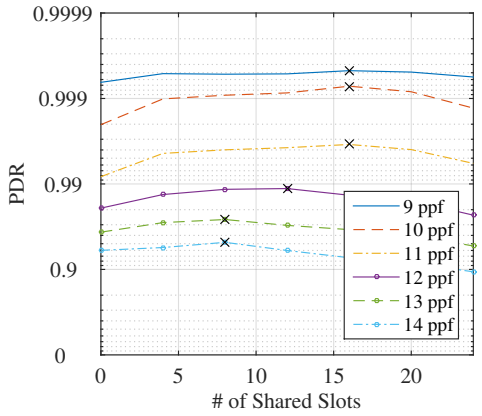


Figure 7: Slot sharing improves the reliability of a TSCH network *on average*. The optimal number of shared slots (marked by the $\times$ symbol) depends on the packet generation rate (packets per frame, ppf) per node. Each point shows the average PDR of 1000 simulation experiments with random link qualities in $[0.5, 1]$.

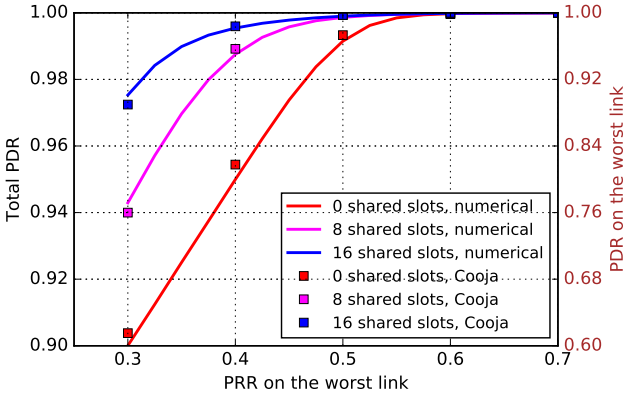


Figure 8: Comparative results for numerical simulations and full-scale simulations in Cooja that demonstrate the benefits of hybrid schedules. The link quality is $PRR = 0.9$ on all links, except for the weakest link (given on the x axis).

16 shared slots yields the best performance for networks with an average PRR of more than 0.8. In TSCH networks with exceptionally good (*i.e.* very high average PRR) and homogeneous links (*i.e* low standard deviation of PRR), all three schedules yield a perfect performance (100% PDR).

In the next series of simulations (Fig. 7), the goal is to identify the optimal number of shared slots in the average case scenario. We assume a constant datarate per end node and compare schedules with a different number of shared slots. For each data point, we report the average PDR over 1000 scenarios with random link qualities (the PRR of each link is selected uniformly in $[0.5, 1]$). In situations where the traffic is close to the TSCH capacity, sharing time slots outperforms a contention-free schedule on average. Nevertheless, the optimum number of shared timeslots decreases as the overall traffic increases.

B. Full-scale implementation and simulation

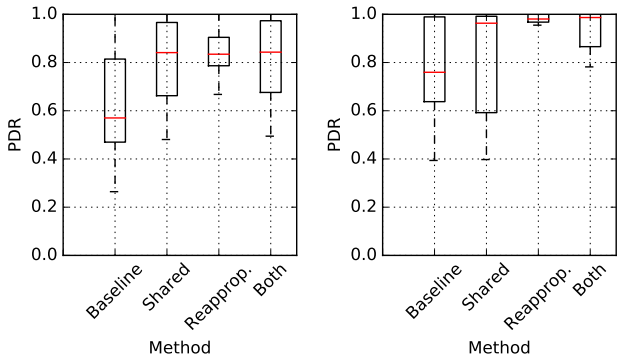
Implementation and simulation setup. We implement the scheduling algorithms (Section V) on top of Contiki, an embedded operating system for the Internet of Things. Contiki features a standard-compatible TSCH implementation [7]. In particular, we add functionality to: (1) dynamically select the channel offset to use for transmissions depending on the target node address, and (2) to support shared slots, used opportunistically with transmission probability depending on queue size. The code is platform-independent and can be run on all platforms on which TSCH is supported.

To evaluate the implementation, we use hardware-emulated Zolertia Z1 nodes (msp430 MCU running @ 8 MHz, CC2420 radio). The TSCH slot duration is set to 30 ms (due to a slower MCU than our CC2650 target platform); other settings are the same as for the numerical simulations.

Results: the star network. We run Cooja simulations for 30 min, generating 120 bytes packets uniformly with rate 10 ppf. In a star network, the results (Fig. 8) show a close match between numerical simulation results and Cooja results (< 0.6 percent point difference in PDR). The results clearly show that the shared slot approach is beneficial as long as link diversity is present. In particular, when the weakest link has $PRR = 0.3$, the improvement in packet error rate is 3.5 times in Cooja (9.62 *vs.* 2.75 % PER) and 4.0 times in the numerical simulations (9.99 *vs.* 2.47 % PER) when comparing between the 0 shared slot and 16 shared slot configurations. The shared slot approach not only improves the overall PDR, but also makes data delivery more fair. All lost packets in this setup are lost on the weakest link; as a result, the shared schedule helps to avoid situations when the data from one node is significantly underrepresented in the experiment's final data set – a situation undesirable for many real-world applications.

Results: multihop networks. We use Cooja to simulate a number of multihop networks with five nodes. In total we simulate five different topologies: two belonging to the class (a) in Figure 5, and three belonging to the class (b) in Figure 5 — *i.e.*, two for which the forwarding data rate is on the average below or equal to the schedule capacity and three where the capacity is overtaxed on one or more links.

The results (Fig. 9) compare four different configuration settings: default TSCH implementation, using the shared slots technique, using slot appropriation, and combining the two techniques. The results show that all of the techniques significantly increase the average PDR. However, when operating at the maximal theoretical capacity (left side of Fig. 9) none of the techniques are capable of achieving $> 80\,\%$ PDR for the least reliable of nodes. This can be explained by the fact that TSCH queue sizes are not infinite. With constant data rate and random packet loss, some burstiness of retransmissions is expected to be present. Whenever such a burst overflows the queue, packets are lost. This behavior is exacerbated by multihop, where bursty losses



(a) $ppf = 8$ **(b)** $ppf = 6$

Figure 9: Slot reappropriation demonstrates better performance in multihop networks. The figure compares baseline TSCH with the shared slot approach, slot reappropriation, and a combination of the two techniques. $PRR = 0.8$ on all links. Left-hand side shows results from a fully saturated schedule, right-hand side: results from a schedule with 25 % overallocation for expected traffic rate. Boxplots from the PDR results of individual nodes; red lines show the median results.

are more likely to be present on forwarding nodes. The Contiki implementation is vulnerable as it uses a relatively small queue size of 8 packets; this number is bounded by the strictly limited RAM size on sensor nodes.

Over-allocating the schedule by 25 % compared to the average-case data rate leads to better results (left side of Fig. 9). In particular, the slot reappropriation technique achieves 98.1 % average PDR and 95.5 % worst-case PDR (for an individual node). The results also show that combining reappropriation and slot sharing techniques gives no additional benefits and has detrimental effect on the worst-case performance because of collisions.

In sum, the reappropriation technique is better for multihop networks even when the number of nodes is small. The slot sharing technique is recommended only if the maximal number of hops cannot be larger than two.

VII. Conclusion

Driven by a real-world deployment in 100 residential environments, in this paper we are dealing with the problem of scheduling high rate unpredictable traffic in IEEE 802.15.4 TSCH networks that operate close to their maximum capacity. Focusing on dynamic environments where reactive scheduling is impractical, we propose an autonomous scheduling methodology which combines: (i) a mixture of dedicated contention-free slots and shared contention-based slots, and (ii) an algorithm that in multihop topologies reappropriates scheduled slots to a different node. We implement our scheduling methodology for the Contiki OS and evaluate it using both numerical and Cooja simulations. The results show that in unpredictable environments the proposed techniques outperform a schedule with dedicated-only slots, improving the average reliability of the TSCH network.

Acknowledgments

This work was performed under the SPHERE IRC funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

[1] "IEEE Standard for Local and metropolitan area networks—Part 15.4," IEEE Std 802.15.42015, 2015.

[2] T. Zhu, Z. Zhong, T. He, and Z.-L. Zhang, "Exploring link correlation for efficient flooding in wireless sensor networks," in *NSDI*, vol. 10, 2010, pp. 1–15.

[3] L. Tang, K.-C. Wang, Y. Huang, and F. Gu, "Channel characterization and link quality assessment of IEEE 802.15.4-compliant radio for factory environments," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 2, pp. 99–110, 2007.

[4] A. Elsts, H. Wennerström, and C. Rohner, "IEEE 802.15.4 Channel Diversity in an Outdoor Environment," in *ACM RealWSN*, 2015.

[5] "IPv6 over the TSCH mode of IEEE 802.15.4e IETF working group," https://tools.ietf.org/wg/6tisch/.

[6] "Minimal 6TiSCH Configuration," https://tools.ietf.org/html/draft-ietf-6tisch-minimal.

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

[8] I. Juc, O. Alphand, R. Guizzetti, M. Favre, and A. Duda, "Energy Consumption and Performance of IEEE 802.15.4e TSCH and DSME," in *IEEE Wireless Commun. and Networking Conf. (WCNC)*, 2016.

[9] A. Elsts, S. Duquennoy, X. Fafoutis, G. Oikonomou, R. Piechocki, and I. Craddock, "Microsecond-Accuracy Time Synchronization Using the IEEE 802.15.4 TSCH Protocol," in *IEEE SenseApp*, 2016.

[10] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15.4e networks," in *IEEE PIMRC*, 2012, pp. 327–332.

[11] 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 J.*, vol. 13, no. 10, pp. 3655–66, 2013.

[12] A. Tinka, T. Watteyne, and K. Pister, "A decentralized scheduling algorithm for time synchronized channel hopping," in *Int. Conf. Ad Hoc Networks*, 2010, pp. 201–216.

[13] M. Hashimoto, N. Wakamiya, M. Murata, Y. Kawamoto, and K. Fukui, "End-to-end reliability-and delay-aware scheduling with slot sharing for wireless sensor networks," in *8th Int. Conf. Commun. Syst. and Networks (COMSNETS)*, 2016.

[14] P. Woznowski *et al.*, "SPHERE: A Sensor Platform for Healthcare in a Residential Environment," in *Designing, Developing, and Facilitating Smart Cities: Urban Design to IoT Solutions*. Springer, 2017, pp. 315–333.

[15] X. Fafoutis *et al.*, "SPW-1: A Low-Maintenance Wearable Activity Tracker for Residential Monitoring and Healthcare Applications," in *Int. Summit on eHealth (eHealth 360)*, 2016, pp. 294–305.

[16] N. Twomey, S. Faul, and W. P. Marnane, "Comparison of accelerometer-based energy expenditure estimation algorithms," in *4th Int. Conf. Pervasive Computing Technologies for Healthcare*. IEEE, 2010, pp. 1–8.

[17] A. Elsts, X. Fafoutis, R. Piechocki, and I. Craddock, "Adaptive Channel Selection in IEEE 802.15.4 TSCH Networks," in *1st Global Internet of Things Summit (GIoTS)*, 2017.