# Adaptive Static Scheduling in IEEE 802.15.4 TSCH Networks

Xenofon Fafoutis, Atis Elsts, George Oikonomou, Robert Piechocki and Ian Craddock
Department of Electrical and Electronic Engineering
University of Bristol, UK
Email: {xenofon.fafoutis, atis.elsts, g.oikonomou, r.j.piechocki, ian.craddock}@bristol.ac.uk

***Abstract*—TSCH (Time-Slotted Channel Hopping) is a synchronous MAC (Medium Access Control) protocol, introduced with the recent amendments to the IEEE 802.15.4 standard. Due to its channel hopping nature, TSCH is a promising enabling technology for dependable IoT (Internet of Things) infrastructures that are deployed in environments that are prone to interference. In TSCH, medium access is orchestrated by a schedule that is distributed to all the nodes in the network. In this paper, we propose Adaptive Static Scheduling to improve the energy efficiency of TSCH networks. Adaptive Static Scheduling builds on top of static schedules and allows each pair of communicating nodes to adaptively activate a subset of their allocated slots, effectively reducing the idle listening overhead of unused slots. Moreover, the nodes can dynamically activate more slots when they need to support bursts of high traffic, without the need of redistributing new schedules. Simulation results demonstrate that Adaptive Static Scheduling outperforms static scheduling in dynamic environments, operating nearly as efficiently as an oracle with knowledge of the optimal schedule.**

***Index Terms*—Time-Slotted Channel Hopping, Scheduling, IEEE 802.15.4-2015, Internet of Things**

## I. Introduction

The recent amendments to the IEEE 802.15.4 standard introduced TSCH (Time-Slotted Channel Hopping) [1], a synchronous MAC (Medium Access Control) protocol for duty-cycling low power networks. TSCH has attracted significant attention from the research community as it promises more reliable and predictable wireless networking for the Internet of Things (IoT) [2]. Indeed, TSCH constitutes a promising enabling technology for dependable IoT infrastructures that are deployed in interference-prone environments, as it implements channel hopping at the link layer. Additionally, its time-slotted nature makes the protocol relatively predictable, in terms of energy consumption and delay. These characteristics are generally considered particularly desirable in critical IoT applications. As a result, TSCH has been implemented for several IoT operating systems (see *e.g.*, the OpenWSN [3] and the Contiki OS [4] implementations) and used in several IoT domains, such as eHealth [5], Smart Grid [6] and Industrial IoT [7]. In addition, there are standardisation efforts for the use of IPv6 over TSCH by the 6TiSCH IETF working group [8]; efforts that can enable the Internet of Everything [9].

TSCH combines elements of TDMA (Time Division Multiple Access) and FDMA (Frequency Division Multiple Access). Indeed, time in TSCH is globally synchronised and divided in timeslots. Access to the medium is controlled by a global schedule which allocates slots to particular wireless links. A TSCH schedule can be visualised as a matrix where each row corresponds to a channel offset and each column corresponds to a time offset. For each transmission, the physical channel is determined pseudo-randomly, combining the channel offset in the schedule and the `ASN` (Absolute Slot Number), *i.e.* a globally-synchronised counter of the slots that have passed since the beginning of the TSCH network. The TSCH schedule is executed within a TSCH frame and is repeated perpetually. Thus, the duration of the frame is equal to the length of the TSCH schedule.

In this paper, we focus on increasing the energy efficiency of TSCH networks that are characterised by dynamic traffic: either because of dynamic traffic generation rates from the sensor nodes or because of the ever-changing wireless channel conditions and, thus, the unpredictable number of retransmissions. The timescale of these changes is often in the order of seconds or even milliseconds. In such scenarios, it is impractical to generate and distribute new schedules in a reactive manner. Instead, static scheduling with proactive overallocation of TSCH slots is necessary.

The number of allocated TSCH slots in a static TSCH schedule controls a performance trade-off. On one hand, conservative overallocation may lead to packet loss as the capacity of the TSCH schedule is not sufficient to support bursts of high traffic. On the other hand, excessive overallocation of TSCH slots can hinder the energy efficiency of the TSCH network, since each allocated slot that is not used by the sender, translates to idle listening overheads for the receiver.

In this paper, we propose Adaptive Static Scheduling to address this challenge. The proposed scheduling scheme builds on top of static TSCH schedules with excessive overallocation, allowing the owners of the timeslots (sender and receiver) to adapt the number of the allocated slots they wish to use, based on the level of slot utilisation. Adaptive Static Scheduling is evaluated using a discrete-time TSCH simulator [10] that is extended to capture the aforementioned trade-off. The evaluation results demonstrate that (i) Adaptive Static Scheduling outperforms static scheduling in environments with dynamic traffic conditions; and (ii) the performance of Adaptive Static Scheduling is very near to the optimal (yet unreachable) performance of an *oracle* schedule.

978-1-5090-4130-5/16/$31.00 

The remainder of the paper is structured as follows. Section II summarises the related work. Section III quantifies the energy efficiency trade-off of static scheduling. Section IV elaborates on the details of Adaptive Static Scheduling. Section V evaluates the proposed scheme. Lastly, Section VI concludes the paper.

## II. Background

### A. Related Work

According to the TSCH specification [1], time in a TSCH network is globally synchronised and divided in *timeslots*. The typical duration of a TSCH timeslot is 10 ms. Within a single active timeslot, transmission or reception of a single packet along its acknowledgement takes place. Timeslots are grouped in *slotframes*; a single slotframe consists of a number of cells, described by timeslot, channel offset, type (e.g., Tx, Rx, or idle), and other properties, such as whether the timeslot is dedicated to a single pair of nodes, or shared. The *schedule* of a TSCH network is a collection of one or more periodically repeating slotframes.

The currently active 6tisch IETF working group [8] is working on integrating TSCH with the low-power IPv6 (6LoWPAN) network stack. The minimal 6tisch schedule [11] proposed by this working group consists of a single slotframe with a single, shared active slot. The main goals of this schedule are interoperability, flexibility, and basic fall-back functionality for the moments when a more complicated schedule is not present. This minimal schedule is designed to support event-driven (unpredictable) traffic, and does not require knowledge of the topology of the network. There are two main weaknesses of the slotted-Aloha-like behaviour of the 6tisch minimal schedule: a large number of collisions when the traffic rate is high, and suboptimal energy usage due to idle listening when it is low.

Duquennoy *et al.* [12] propose Orchestra, a scheduling mechanism exploiting the state of RPL routing in 6tisch networks. A node running the Orchestra schedule allocates slots for traffic of its single parent node and each of its child nodes; the slots can be either dedicated or shared. However, in contrast to this work, Orchestra is not adaptive with regards to the traffic rate.

Elsts *et al.* [10] propose a schedule for high rate traffic. Here, some dedicated slots between each pair of nodes coexist with shared slots from the nodes to the gateway in the same slotframe. This schedule adds extra flexibility compared with a purely dedicated schedule, as nodes with worse links or more data to send are able to improve their performance by using the shared slots. However, [10] assumes permanently-on nodes and does not concern itself with energy efficiency – a challenge we aim to handle in the present paper.

There are plenty of existing centralized TSCH scheduling algorithms. For a few examples, Palattella *et al.* [13] use graph colouring methods to find the optimal schedule of a TSCH network. Exarchakos *et al.* [14] in describe a web service for adaptive scheduling of TSCH networks based on inter-arrival time of packets. However, the additional requirement of a centralised monitoring and control service leads to a significant increase in implementation and run-time complexity and adds overhead.

Tinka *et al.* [15] propose a decentralised, reservation based scheduling algorithm. Their work is tailored towards highly dynamic networks with mobile nodes, and their scheduling scheme spent a lot of resources to discover and maintain the set of neighbours of a node – a functionality and overhead not necessary for this work that deals with statically deployed networks.

### B. Motivation

This work is motivated by the SPHERE (a Sensor Platform for HEalthcare in a Residential Environment) deployment in 100 houses in Bristol, UK [16]. SPHERE is a multi-purpose sensing platform that is using several sensing modalities to capture health-related data, including: (i) a low-power environmental sensor network, which captures light, humidity, temperature, presence, and noise sensors at a room-level granularity [17]; and (ii) a network of low-power wearable devices equipped with accelerometer sensors (one per home resident) broadcasting multiple packets per second [18]. A TSCH home network, based on Contiki TSCH [4], is responsible for relaying the sensor data to a central station for post-processing and long-term storage.

The TSCH networks of SPHERE rely on static scheduling with overallocation, because of the fact that they are deployed in very dynamic environments in which the reactive generation and distribution of TSCH schedules is impractical. Indeed, residential deployments are very dynamic because of the unpredictable human factor. For example, (i) the amount of traffic a TSCH node needs to relay is ever-changing due to the mobility of wearable sensors; (ii) unpredictable body shadowing is responsible for dynamic link qualities and retransmissions; and (iii) electromagnetic interference is generated by the residents of the house (*e.g.* laptops, smart TVs, etc.). The level of slot overallocation in static TSCH schedules controls a performance trade-off. On one hand, modest overallocation cannot provide the necessary capacity for bursts of high traffic, leading to packet loss. On the other the hand, excessive overallocation introduces idle listening energy overheads at the receiver. Adaptive Static Scheduling, proposed in this paper, handles this trade-off, providing a means of improving the energy efficiency of TSCH networks that depend on static scheduling, such the TSCH networks of SPHERE.

## III. The Energy Efficiency of TSCH Networks

### A. The TSCH Simulator

To evaluate the energy efficiency of TSCH networks, we employ and extend the discrete-time event TSCH Simulator[1] [10]. This tool simulates star-type TSCH neighbourhoods with arbitrary TSCH schedules. For each TSCH link, the simulator supports retransmissions, a configurable queue size, and a configurable link-layer packet reception probability, denoted

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

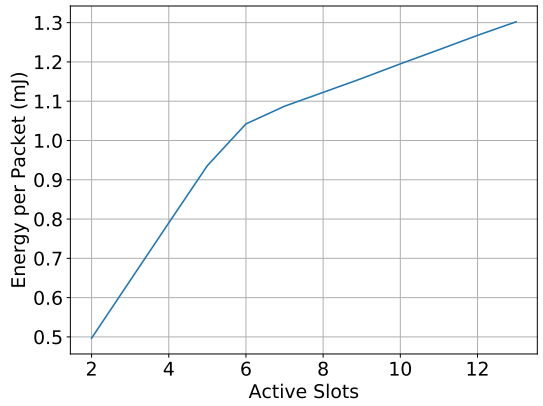


Fig. 1. Energy consumed per packet for various numbers of active slots per sender per frame ($r = 4$, $p = 0.7$).

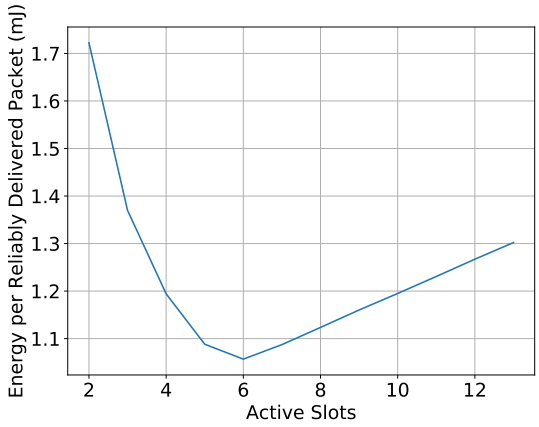


Fig. 3. Energy per reliably delivered packet (Equation 5) for various numbers of active slots per sender per frame ($r = 4$, $p = 0.7$).

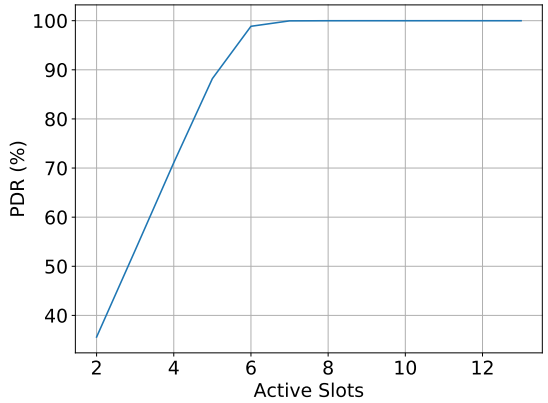


Fig. 2. Packet delivery rate (with retransmissions) for various numbers of active slots per sender per frame ($r = 4$, $p = 0.7$).

as $p$, that models the effect of channel errors and interference. The simulator supports both dedicated (contention-free) slots and shared (contention-based) slots; yet, in this work we use solely contention-free schedules.

The original simulator evaluates different schedules and assesses their quality. Indeed, it estimates the end-to-end Packet Delivery Rate (PDR) via keeping track of the number of packets that are lost either due to full queues or because the maximum number of retransmissions is reached. For the purposes of this work, we extend the simulator with the capability of estimating the energy consumed for the execution of the schedule. Focusing on contention-free schedules with unicast transmissions each slot can be classified in one of the following three categories:

- *Sleeping slot*: This slot is specified as unused in the TSCH schedule. Both the sender and the receiver are sleeping with the radios turned off.
- *TxRx slot*: This slot is dedicated for communication between a sender and a receiver. Hence, the sender wakes up with the intention to transmit and the receiver listens to the channel for incoming transmissions. In addition, the queue of the sender has at least one frame to transmit to the sender.
- *Idle listening slot*: This slot is dedicated for communication between a sender and a receiver. However, in this case, the queue of the sender is empty. As a result, the receiver wastes energy in idle listening.

We have extended the simulator to keep track of the number of slots that fall in each category during the simulation, namely $N_{sleep}$, $N_{txrx}$ and $N_{idle}$, respectively. The total energy consumed during the sleeping slots can be calculated by:

$$E_{sleep} = N_{sleep} \cdot (2 \cdot Q_{sleep} \cdot V) \ , \tag{1}$$

where $V$ is the supply voltage of the system, $Q_{sleep}$ is the electric charge consumed during a sleeping slot (given a supply voltage $V$) and the factor 2 corresponds to the sender and the receiver. Similarly, the total energy consumed during the TxRx slots can be calculated by:

$$E_{txrx} = N_{txrx} \cdot (Q_{TxDataRxAck} \cdot V + Q_{RxDataTxAck} \cdot V) \ , \tag{2}$$

where $Q_{TxDataRxAck}$ is the electric charge consumed by the sender for transmitting a data packet and receiving an acknowledgement and $Q_{RxDataTxAck}$ is the electric charge consumed by the receiver respectively. These values also represent an upper bound for the case of an a unsuccessful data exchange due to channel errors or interference. For simplicity, we treat these events similarly to successful transmissions and count them as part of $N_{txrx}$. Lastly, the total energy consumed during the idle listening slots can be calculated by:

$$E_{idle} = N_{idle} \cdot (Q_{sleep} \cdot V + Q_{idle} \cdot V) \ , \tag{3}$$

where $Q_{idle}$ is the electric charge consumed by the receiver for idle listening. For the electric charge values we use the measurements on the GINA mote [19], presented in [20] and

Table I (supply voltage $V = 3.3$ V). The average energy per packet is estimated by:

$$E = \frac{E_{sleep} + E_{txrx} + E_{idle}}{r \cdot \texttt{FRAMES}} \ , \tag{4}$$

where $r$ the traffic rate expressed in packets per frame and FRAMES is the total number of simulated frames.

TABLE I
MEASURED CHARGE DRAWN FOR EACH TYPE OF SLOT [20]

| State | Charge ($\mu$C) |
|---|---|
| $Q_{sleep}$ | 4.9 |
| $Q_{TxDataRxAck}$ | 92.6 |
| $Q_{RxDataTxAck}$ | 96.3 |
| $Q_{Idle}$ | 47.9 |

### B. The Energy Efficiency Trade-off

Using our TSCH simulator, in this section, we demonstrate the energy efficiency trade-off of static schedules and define an performance metric that evaluates jointly both the reliability and the energy consumption of TSCH networks.

Let us simulate a TSCH neighbourhood which consists of 4 senders and a single receiver node. Let us also assume that the frame size is 100 slots, the maximum queue size is equal to 8 packets, the maximum retransmission attempts is 8, each link has a packet reception probability of $p = 0.7$, and the traffic rate is constant, $r = 4$ packets per frame. Fig. 1 and Fig. 2 plot the average energy consumed per packet and the reliability (PDR) of the TSCH network respectively, for various numbers of active slots per sender per frame. In both graphs, we can clearly observe a trend shift at 6 active slots, which roughly corresponds to the Expected Transmission Count (ETX), $r/p = 5.71$. Reducing the active slots, is beneficial for the energy consumption as the radios spend more time in sleeping. Yet, if the schedule has less than 6 active slots, the capacity of the schedule is not sufficient to relay all the traffic. On the other hand, if the schedule has more than 6 active slots, energy is wasted in idle listening. To capture this trade-off we define $\eta$ as the energy consumed per reliably delivered packet, similarly to [21]:

$$\eta = \frac{E}{\texttt{PDR}^n} \ , \tag{5}$$

where $n$ controls the relative importance of reliability against energy consumption. The lower the $\eta$ metric, the more energy-efficient the TSCH network is. Fig. 3 plots $\eta$ for the above scenario ($n = 1.2$). The simulations demonstrate the fact that there exists an optimum schedule (in this case, 6 active slots) that maximises the energy efficiency of the TSCH network.

## IV. ADAPTIVE STATIC SCHEDULING

A properly configured static schedule would perform optimally in a static scenario. However, a fixed number of active slots would perform sub-optimally when either the traffic or the channel conditions are dynamic. In this paper, we propose Adaptive Static Scheduling to address this issue.

TABLE II
AN ADAPTIVE STATIC SCHEDULE WITH $S_a = 3$ AND $S_m = 4$

| | | | | | |
|---|---|---|---|---|---|
| A → B | | | A → B | | |
| | | | | | |
| | | A → B | | | |
| | | | | | |
| | | | | | A → B |

Notes: '$x \rightarrow y$' correspond to a contention-free slot allocated to sender $x$ and receiver $y$. Gray slots are activated allocated slots, whereas white slots are inactive allocated slots.

Adaptive Static Scheduling builds on top of static schedules, inheriting their simplicity, robustness and efficiency. The scheme operates as follows. A static schedule allocates $S_m$ dedicated (contention-free) slots per frame to a particular pair of nodes (sender and receiver). $S_m$ should be high enough to support the worst-case scenario traffic conditions. On top of this static schedule, the sender and the receiver adaptively agree on the number of slots that they wish to use, denoted as $S_a \in [1, S_m]$. In other words, the first $S_a$ slots are activated and the remaining $S_m - S_a$ of their allocated slots are inactive (example shown in Table II). The sender maintains a Exponentially Weighted Moving Average (EWMA) of the utilisation level of the activated slots ($u$) with coefficient $\alpha$. Before a transmission, the sender adapts $S_a$ based on $u$ and a particular policy, and piggy-backs the updated value in the header of the packet. Note that this introduces a negligible overhead of $log_2(S_m)$ bits. If the packet transmission is successful, the sender and the receiver adapt the schedule to reflect the current $S_a$ value. The process is summarised in Algorithm 1.

**Algorithm 1** Adaptive Static Scheduling

```
u = u_0;
for every activated slot in the static schedule do
    if QUEUESIZE == 0 then
        u = (1 − α) · u + α · 0;
    else
        u = (1 − α) · u + α · 1;
        adapt S_a by executing Algorithm 2;
        transmit packet including S_a in the header;
        if transmission acknowledged then
            apply adaptation S_a;
        end
    end
end
```

Algorithm 2 summarises the adaptation policy which is characterised by a low and high threshold: $u_l$ and $u_h$. Note that we do not decrease the number of activated slots unless

**Algorithm 2** Adaptation Policy

```
input : S_a, u
output: S_a
if u > u_h then
    S_a = min(S_m, S_a + 1);
end
if u < u_l and QUEUESIZE == 1 then
    S_a = max(1, S_a − 1);
end
```

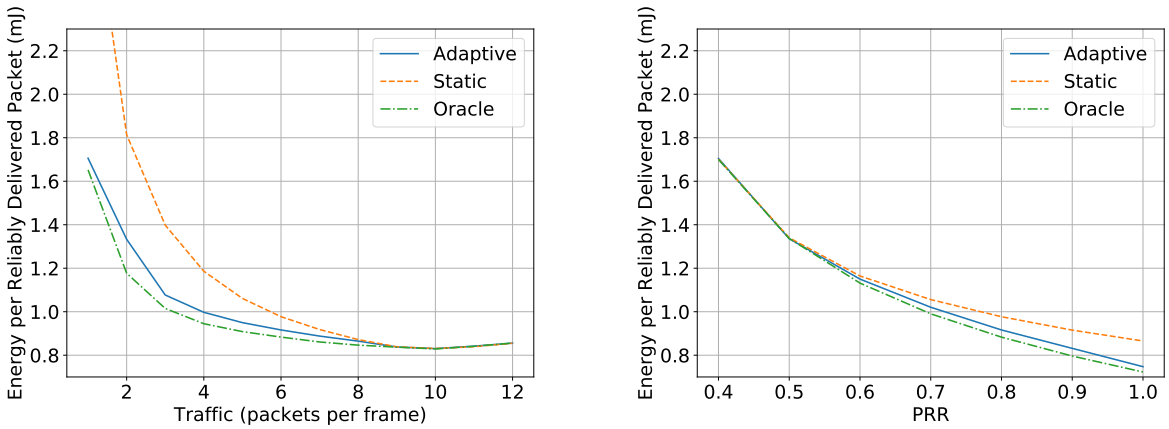


Fig. 4. The energy per reliably delivered packet (Equation 5) of Adaptive Static Scheduling compared against a static schedule that favours ***reliability*** and an oracle schedule. Left: depending on traffic rate; right: depending on channel quality.

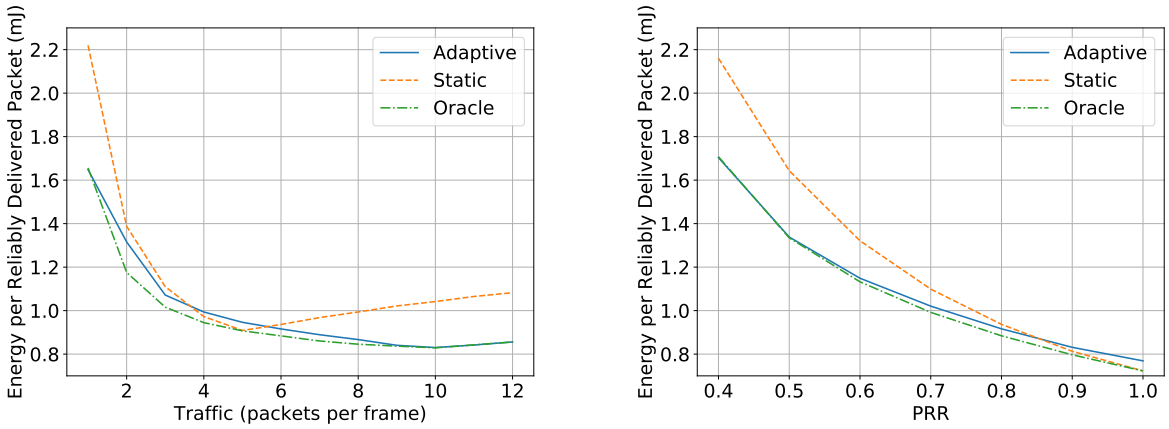


Fig. 5. The energy per reliably delivered packet (Equation 5) of Adaptive Static Scheduling compared against a static schedule that favours ***energy consumption*** and an oracle schedule. Left: depending on traffic rate; right: depending on channel quality.

the queue size is expected to be empty after the transmission of the current packet, regardless of the slot utilisation level.

It should be noted that Adaptive Static Scheduling is not applicable on busy TSCH networks that relay very high traffic. Indeed, in such scenarios TSCH slots are a scarce resource and it would be inefficient to keep them allocated when inactive. For those scenarios, the authors recommend alternative scheduling techniques [10].

## V. Evaluation

We evaluate Adaptive Static Scheduling by comparing it against a static schedule that favours reliability against energy consumption (*i.e.* $S_a$ always equal to $S_m$), a static schedule that favours energy consumption against reliability (*i.e.* $S_a$ always equal to $S_m/2$), and an *oracle* static schedule in which we brute-force all possible static schedules and select the best. The simulations are conducted using the TSCH simulator (see Section III-A) with the configuration parameters listed in Table III. Each simulation is repeated 100 times and the average values are reported. All standard deviations are less than 3% of the reported average values.

TABLE III
Simulation Parameters

| Parameter | Notation | Value |
|---|---|---|
| Frame Size | - | 100 slots |
| Duration of Simulation | FRAMES | 100 frames |
| Max. Retransmissions | - | 8 |
| Max. Queue Size | - | 8 |
| No. of Sender Nodes | - | 4 |
| Allocated Slots | $S_m$ | 12 |
| EWMA coefficient | $\alpha$ | 0.1 |
| Initial utilisation level | $u_0$ | 0.95 |
| High utilisation threshold | $u_h$ | 0.9 |
| Low utilisation threshold | $u_l$ | 0.8 |
| Energy Efficiency Exponent | $n$ | 1.2 |

In the first series of simulations, we consider dynamic traffic ($r$) under static channel conditions ($p = 0.8$). Fig. 4a plots the energy per reliably delivered packet of the TSCH network (Eq. 5) under various traffic conditions ($r \in [1, 12]$

packets per frame). In this simulation, the static schedule is configured to favour reliability against energy consumption. As a result, the static schedule performs optimally in high traffic conditions ($r > 8$). However, in low traffic conditions, the static schedule is highly inefficient, up to two times more energy per reliably delivered packet compared to the oracle. Adaptive Static Scheduling, on the other hand, adapts to all the traffic conditions, performing near-optimally. The simulation in Fig. 5a is similar except for the fact that the static schedule is configured to favour energy consumption instead of reliability. It can be observed that static schedule operates optimally only when the traffic conditions match it ($r = 5$), similarly to Fig. 3. However, it is inefficient otherwise. Again, the energy efficiency of Adaptive Static Scheduling is very close to the optimal performance of the oracle.

In the second series of simulations, we consider dynamic channel conditions ($p$) under static traffic ($r = 6$). First, we consider the static schedule that favours reliability instead of energy consumption. Fig. 4b plots the energy per reliably delivered packet of the TSCH network (Eq. 5) under various link-layer PRRs ($p \in [0.4, 1.0]$). Similarly to the previous simulations, the static schedule performs optimally in very bad channel conditions, yet inefficiently in good channel conditions. The performance of Adaptive Static Scheduling, on the other hand, converges to a near-optimal configuration. This trend is also verified in the last simulation, in which the static schedule now favours energy consumption.

Overall, the simulations demonstrate that Adaptive Static Scheduling is a robust scheduling solution that achieves near-optimal performance in a wide variety of dynamic scenarios.

## VI. Conclusion

This work focuses on TSCH networks that rely on static scheduling with overallocation, because of the fact that they are deployed in very dynamic environments in which the reactive generation and distribution of TSCH schedules is impractical. The level of slot overallocation in TSCH schedules, however, controls a performance trade-off between reliability and energy consumption. In this paper, we propose Adaptive Static Scheduling to allow each pair of nodes to adaptively manage their allocated slots in a fully distributed manner. Practically, Adaptive Static Scheduling builds on top of static schedules with excessive slot overallocation; yet, the nodes can dynamically activate or deactivate their allocated slots, matching them to the current traffic conditions. The proposed scheme is compared against static scheduling in a series of simulations. The results demonstrate that Adaptive Static Scheduling operates robustly in a wide variety of application-layer traffic levels and wireless channel conditions, reaching near-optimal energy efficiency levels.

## Acknowledgment

This work was performed under the SPHERE (a Sensor Platform for HEalthcare in a Residential Environment) 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] D. Singh, G. Tripathi, and A. J. Jara, "A survey of Internet-of-Things: Future vision, architecture, challenges and services," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, March 2014, pp. 287–292.

[3] T. Watteyne, X. Vilajosana, B. Kerkez *et al.*, "OpenWSN: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[4] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *13th Int. Conf. on Distributed Comput. in Sensor Syst. (DCOSS)*, 2017.

[5] A. Elsts, G. Oikonomou, X. Fafoutis, and R. Piechocki, "Internet of Things for smart homes: Lessons learned from the SPHERE case study," in *Global Internet of Things Summit (GIoTS)*, June 2017, pp. 1–6.

[6] A. Paventhan, B. D. Darshini, H. Krishna, N. Pahuja, M. F. Khan, and A. Jain, "Experimental evaluation of IETF 6TiSCH in the context of Smart Grid," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 530–535.

[7] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: deterministic IP-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, December 2014.

[8] P. Thubert, T. Watteyne, M. R. Palattella, X. Vilajosana, and Q. Wang, "IETF 6TSCH: Combining IPv6 Connectivity with Industrial Performance," in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July 2013, pp. 541–546.

[9] A. J. Jara, L. Ladid, and A. Skarmeta, "The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 4, no. 3, pp. 97–118, 9 2013.

[10] A. Elsts, X. Fafoutis, J. Pope, G. Oikonomou, R. Piechocki, and I. Craddock, "Scheduling High-Rate Unpredictable Traffic in IEEE 802.15.4 TSCH Networks," in *13th Int. Conf. on Distributed Comput. in Sensor Syst. (DCOSS)*, 2017.

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

[12] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 337–350.

[13] 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 *Proc. IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun. (PIMRC)*, 2012, pp. 327–332.

[14] 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*, 2016.

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

[16] P. Woznowski, A. Burrows, T. Diethe *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 International Publishing, 2017, pp. 315–333.

[17] X. Fafoutis, A. Elsts, A. Vafeas, G. Oikonomou, and R. Piechocki, "Demo: SPES-2 – A Sensing Platform for Maintenance-Free Residential Monitoring," in *Proc. Int. Conf. Embedded Wireless Systems and Networks (EWSN)*, 2017, pp. 240–241.

[18] X. Fafoutis, A. Vafeas, B. Janko *et al.*, "Designing Wearable Sensing Platforms for Healthcare in a Residential Environment," *EAI Endorsed Trans. Pervasive Health and Technology*, vol. 17, no. 12, Sept. 2017.

[19] A. M. Mehta and K. S. J. Pister, "WARPWING: A complete open source control platform for miniature robots," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 5169–5174.

[20] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. S. J. Pister, "A Realistic Energy Consumption Model for TSCH Networks," *IEEE Sensors Journal*, vol. 14, no. 2, pp. 482–489, Feb 2014.

[21] G. Z. Papadopoulos, A. Mavromatis, X. Fafoutis, R. Piechocki, T. Tryfonas, and G. Oikonomou, "Guard Time Optimisation for Energy Efficiency in IEEE 802.15.4-2015 TSCH Links," in *Proc. 2rd EAI Int. Conf. on Interoperability in IoT (Inter-IoT)*, 2016, pp. 56–63.